# Contributions à la modélisation et la conception des systèmes de gestion de provenance à large échelle

Mohamed Amin Sakka


TELECOM SudParis


Université d'Évry Val d'Essonne

**T**hèse de doctorat de Télécom SudParis, préparée dans le cadre de l'école doctorale S&I, en accréditation conjointe avec l'Université d'Evry-Val d'Essonne

*Spécialité Informatique*

Par :

**M**ohamed Amin SAKKA

# **C**ontributions à la modélisation et la conception de systèmes de gestion de provenance à large échelle

**T**hèse présentée pour l'obtention du grade de
Docteur de TÉLÉCOM SudParis

**S**outenue le 28 Septembre 2012 devant le jury composé de :

*Rapporteurs :*

| | | |
|---|---|---|
| Mme. | Christine COLLET | Professeur à l'ENSIMAG (Grenoble INP) - LIG |
| M. | Omar BOUCELMA | Professeur à l'université de Marseille Nord - LSIS |

*Examinateurs :*

| | | |
|---|---|---|
| M. | Bruno DEFUDE | Professeur à TÉLÉCOM SudParis - SAMOVAR (Directeur de thèse) |
| M. | Jorge Luis TELLEZ | Docteur en Informatique, Président & Fondateur QualyCloud (Encadrant) |
| Mme. | Amel BOUZEGHOUB | Professeur à TÉLÉCOM SudParis - SAMOVAR |
| M. | Bernd AMANN | Professeur à l'université Pierre et Marie Curie - LIP6 |

Thèse numéro : 2012TELE0023

# Remerciements

Déjà plus de trois ans depuis le tout début. Pendant ces trois ans, j'ai rencontré des femmes et des hommes qui ont contribué à ce projet et à qui j'adresse mes remerciements les plus sincères.

Je souhaite en premier lieu remercier les membres du jury, Christine COLLET, Omar BOUCELMA, Amel BOUZEGHOUB et Bernd AMANN pour avoir accepté d'examiner mon travail.

Je remercie Bruno DEFUDE, mon encadrant et directeur de thèse qui m'a accueilli à bras ouverts dans le laboratoire SAMOVAR. Avec lui, j'ai apprécié une rigueur de travail, des discussions souvent fructueuses et surtout un support scientifique inconditionné. En dehors de son apport scientifique, je n'oublierai pas de le remercier également pour ses qualités humaines et son soutien qui m'ont permis de mener à bien cet ouvrage.

Je tiens à exprimer ma gratitude à mon encadrant en entreprise Jorge Luis TELLEZ. Je le remercie pour la précieuse confiance qu'il m'a donnée, pour la grande liberté d'idées et de travail qu'il m'a accordée et pour le temps qu'il m'a consacré.

Mes remerciements vont aussi à Novapost, l'entreprise qui a financé en grande partie mon travail de thèse. Je remercie également mes ancien collègues chez Novapost : Yann, Vincent, Pascal, Maxime, Julie, Céline, Silvère, Benoît et Benjamin.

Je remercie les membres de l'équipe SIMBAD, en particulier Samir TATA et Walid GAALOUL. Je remercie aussi ceux que j'ai côtoyés et qui, de loin ou de près, ont apporté une plus-value à la qualité de mon travail par leurs conseils, leurs remarques et leurs observations.

Je tiens également à remercier mes anciens et actuels collègues du laboratoire : Mohamed, Mohamed (2), Amine, Chan, Leon, Rami, Walid, Hichem, Imene, Dorsaf et Olfa. C'était très agréable de passer ces années en leur compagnie.

Je tiens spécialement à remercier ma famille : mes parents et mes sœurs qui, malgré l'éloignement, ont cru à mon projet, m'ont toujours soutenu et donné leur confiance. Je leur suis reconnaissant pour les sacrifices qu'ils ont du faire pendant mes longues années d'études et d'absence.

Merci à mes chers amis de toujours : Ahmed, Ghassen, Valentina, Hana, Majed, Raja,

Sofiene, Karim, Bilel, Mohamed, Refka et Kawthar.

Merci à tous ceux qui y ont cru ...

# Résumé

Les avancées dans le monde des réseaux et des services informatiques ont révolutionné les modes d'échange, de partage et de stockage de l'information. Nous migrons de plus en plus vers des échanges numériques ce qui implique un gain en terme de rapidité de transfert, facilité de partage et d'accès ainsi qu'une efficacité d'organisation et de recherche de l'information. Malgré ses avantages, l'information numérique a l'inconvénient d'être volatile et modifiable ce qui introduit des problèmes liés à sa provenance, son intégrité et sa valeur probante. Dans ce contexte, la provenance apparaît comme une méta-donnée clé qui peut servir pour juger la qualité de l'information et pour vérifier si elle répond à un ensemble d'exigences métier, techniques et légales.

Aujourd'hui, une grande partie des applications et des services qui traitent, échangent et gèrent des documents électroniques sur le web ou dans des environnements Cloud génèrent des données de provenance hétérogènes, décentralisés et non interopérables. L'objectif principal de cette thèse est de proposer des solutions génériques et interopérables pour la modélisation de l'information de provenance et de concevoir des architectures de systèmes de gestion de provenance passant à l'échelle tant au niveau du stockage et que de l'exploitation (interrogation).

Dans la première partie de la thèse, nous nous intéressons à la modélisation de la provenance. Afin de pallier à l'hétérogénéité syntaxique et sémantique qui existe entre les différents modèles de provenance, nous proposons une approche globale et cohérente pour la modélisation de la provenance basée sur les technologies du web sémantique. Notre approche repose sur un modèle de domaine minimal assurant un niveau d'interprétation minimal et commun pour n'importe quelle source de provenance. Ce modèle peut ensuite être spécialisé en plusieurs modèles de domaine pour modéliser des concepts et des propriétés métier différents. Cette spécialisation assure l'interopérabilité sémantique souhaitée et permet par la suite de générer des vues métiers différentes sur les mêmes données de provenance.

Dans la deuxième partie de la thèse, nous nous focalisons sur la conception des systèmes de gestion de provenance (ou PMS). Nous proposons tout d'abord une architecture logique de PMS indépendante des choix technologiques d'implémentation et de déploiement. Cette architecture détaille les modules assurant les fonctionnalités requises par notre approche

de modélisation et sert comme architecture de référence pour la conception d'un PMS. Par la suite, et afin de préserver l'autonomie des sources de provenance, nous proposons une architecture distribuée de PMS à base de médiateur. Ce médiateur a une vision globale sur l'ensemble des sources et possède des capacités de distribution et de traitement de requêtes. Finalement la troisième partie de la thèse valide nos propositions. La validation de notre approche de modélisation a été réalisée dans un cadre industriel chez Novapost, une entreprise proposant des services SaaS pour l'archivage de documents à valeur probante. Ensuite, l'aspect passage à l'échelle de notre architecture a été testé par l'implémentation de deux prototypes de PMS sur deux technologies de stockage différentes : un système RDF (Sesame) et un SGBD NoSQL (CouchDB). Les tests de montée en charge effectués sur les données de provenance Novapost ont montré les limites de Sesame tant pour le stockage que pour l'interrogation en utilisant le langage de requêtes SparQL, alors que la version CouchDB associée à un langage de requêtes basé sur map/reduce a démontré sa capacité à suivre la charge de manière linéaire en augmentant le nombre de serveurs.

**Mots clés :** modélisation de la provenance, système de gestion de provenance, traitement de requêtes distribuées, passage à l'échelle.

# Summary

Provenance is a key metadata for assessing electronic documents trustworthiness. It allows to prove the quality and the reliability of its content. With the maturation of service oriented technologies and Cloud computing, more and more data is exchanged electronically and dematerialization becomes one of the key concepts to cost reduction and efficiency improvement. Although most of the applications exchanging and processing documents on the Web or in the Cloud become provenance aware and provide heterogeneous, decentralized and not interoperable provenance data, most of Provenance Management Systems (PMSs) are either dedicated to a specific application (workflow, database, ...) or a specific data type. Those systems were not conceived to support provenance over distributed and heterogeneous sources. This implies that end-users are faced with different provenance models and different query languages. For these reasons, modeling, collecting and querying provenance across heterogeneous distributed sources is considered today as a challenging task. This is also the case for designing scalable PMSs providing these features.

In the fist part of our thesis, we focus on provenance modelling. We present a new provenance modelling approach based on semantic Web technologies. Our approach allows to import provenance data from heterogeneous sources, to enrich it semantically to obtain high level representation of provenance. It provides syntactic interoperability between those sources based on a minimal domain model (MDM), supports the construction of rich domain models what allows high level representations of provenance while keeping the semantic interoperability. Our modelling approch supports also semantic correlation between different provenance sources and allows the use of a high level semantic query language.

In the second part of our thesis, we focus on the design, implementation and scalability issues of provenance management systems. Based on our modelling approach, we propose a centralized logical architecture for PMSs. Then, we present a mediator based architecture for PMSs aiming to preserve provenance sources distribution. Within this architecture, the mediator has a global vision on all provenance sources and possesses query processing and distribution capabilities.

The validation of our modelling approach was performed in a document archival context

within Novapost, a company offering SaaS services for documents archiving. Also, we propose a non-functional validation aiming to test the scalability of our architecture. This validation is based on two implementation of our PMS : he first uses an RDF triple store (Sesame) and the second a NoSQL DBMS coupled with the map-reduce parallel model (CouchDB). The tests we performed show the limits of Sesame in storing and querying large amounts of provenance data. However, the PMS based on CouchDB showed a good performance and a linear scalability.

**Keywords :**provenance data, Open Provenance Model, semantic provenance management systems, scalability.

*The White Rabbit put on his spectacles. Where Shall I begin, Please your Majesty ? he asked. Begin at the beginning, the King said gravely, and go till you come to the end : then stop.*

Lewis Carroll, " Alice's Adventures in Wonderland"

# Table des matières

*Chapitre 1*

# Introduction générale

*"Savoir ce que tout le monde sait, c'est ne rien savoir. Le savoir commence là où commence ce que le monde ignore."*

Remy de Gourmont, "Promenades philosophiques"

**Notre thèse est :**

*Les données dont nous ne connaissons pas la provenance ne sont pas fiables.*
*Aujourd'hui, une grande partie des applications n'assurent pas la gestion de la provenance de leurs données. Ceci peut remettre en cause leur fiabilité.*
*Les systèmes de gestion de provenance peuvent améliorer cette fiabilité en proposant des fonctionnalités permettant de connaitre la provenance des données et d'assurer leur traçabilité.*

## 1.1 Contexte et problématique

Les avancées dans le monde des services et des réseaux informatiques ont révolutionné les modes d'échange, de partage et de sauvegarde de l'information. Nous migrons de plus en plus vers des échanges numériques ce qui implique un gain en terme de rapidité des transferts, facilité de partage et d'accès ainsi qu'une efficacité d'organisation et de recherche de l'information. Malgré ses avantages, l'information numérique a l'inconvénient d' être volatile et modifiable ce qui introduit des problématiques sur sa provenance, son intégrité et sa valeur probante.

Nous nous intéressons dans notre travail à une forme spécifique de l'information numérique : **le document électronique**. Ce type d'information est de plus en plus utilisé aussi bien dans les entreprises que chez les particuliers. Les documents électronique peuvent être confidentiels (comme les relevés bancaires, les bulletins de salaire, les bilans médicaux...) ou publics (comme les articles sur les blogs, les images partagées sur les réseaux sociaux...).

Avec l'essor de la dématérialisation, les entreprises proposant des plateformes logicielles ou des services de gestion documentaire doivent prouver à leurs clients la provenance de leurs documents afin d'assurer leur valeur probante. Certes, ces plateformes et ces services ont acquis aujourd'hui un bon niveau de maturité et assurent plusieurs types de fonctionnalités (stockage, partage et téléchargement, gestion des droits d'accès...). Cependant, la gestion des documents à caractère confidentiel suscite plusieurs interrogations. En effet, les utilisateurs, se demandent si ces plateformes assurent la confidentialité et l'intégrité de leurs documents. Ils se posent aussi des questions sur la provenance de leurs documents et sur leur valeur probante en cas de conflit. Les utilisateurs se demandent aussi s'il est possible d'obtenir des éléments de preuve sur les actions effectuées sur les documents et les entités qui les ont réalisés.

A la différence de la vérification du caractère authentique d'un document papier qui est un objet statique, la vérification du caractère probant d'un document numérique consiste en l'analyse d'un cycle de vie qui a pu se dérouler de nombreuses années avant la vérification et qui peut être devenu manquant ou invérifiable.

Si le document est manipulé par un système informatique unique, les problématiques de provenance et de valeur probante sont simplifiées. Cependant, l'aspect multi-acteurs et distribué des plateformes et des services manipulant et traitant les documents électroniques complexifie l'obtention d'informations sur leur provenance et leur cycle de vie.

Notre travail de thèse se déroule dans le cadre d'une convention CIFRE avec Novapost [1], une entreprise proposant un service d'archivage à valeur probante ainsi qu'un ensemble de services de distribution, d'archivage légal, de consultation et de téléchargement de documents à caractère confidentiel en mode SaaS [2]. Dans ce contexte, le cycle de vie standard d'un document comprend des étapes de génération, de traitement, de transfert et d'archivage. Sur ce cycle de vie, plusieurs acteurs et entités juridiques, souvent hétérogènes et distribuées interviennent.

Les solutions actuelles pour la gestion de la provenance et de la traçabilité des documents électroniques reposent sur une accumulation de journaux (principalement des fichiers de log et des enregistrements en bases de données) provenant d'entités hétérogènes, multiples et distantes. Ainsi, les informations relatives au cycle de vie d'un document ou d'un flux informationnel sont distillées sur de multiples sources qu'il faut croiser pour reconstituer son historique. Ajoutons à cela la pérennité des entités responsables de la conservation des données sur ces sources qui est également problématique.

La gestion de la provenance des documents électroniques et le reconstruction de leur cycle de vie est devenue aujourd'hui un besoin qui révèle les problématiques suivantes :

– l'hétérogénéité des formats et des structures de données qui peuvent nuire à la constitution de la trace d'un document,

1. www.novapost.fr
2. Software as a Service

- l'hétérogénéité des systèmes manipulant les documents et des politiques de gestion associées,
- la difficulté d'identifier tous les systèmes intervenant dans la chaîne de traitement du document,
- la difficulté d'identifier toutes les sources de provenance concernées par un document : elles sont distribuées sur de multiples systèmes et gérées par différentes entités,
- l'absence d'un identifiant de document global, unique et partagé entre l'ensemble des systèmes intervenants sur sa chaîne de traitement le long de son cycle de vie et permettant de corréler toutes les informations de provenance le concernant,
- l'hétérogénéité sémantique entre des modèles de provenance distribués sur plusieurs sources et concernant un même document,
- la difficulté d'interroger des données de provenance stockées dans des formats non standards (fichiers de logs, formats propriétaires, bases de données...),
- la difficulté pour différents acteurs d'interroger les mêmes données de provenance avec des besoins métier différents,
- l'expressivité limitée des requêtes de provenance (il s'agit de pouvoir formuler des requêtes à valeur métier qui correspondent aux concepts et aux propriétés d'un métier donné).

## 1.2 Problématique de recherche

La gestion de la provenance des documents électroniques est un facteur clé pour assurer leur valeur probante. Pour cela, nous avons besoin de collecter des informations intègres, complètes et riches sur la provenance des documents et de pouvoir les interroger. Cette gestion nécessite de pouvoir modéliser l'information de provenance et de pallier aux problématiques relatives à l'hétérogénéité des sources de provenance à différents niveaux. Ensuite, il s'agit de concevoir des systèmes assurant la gestion de l'information de provenance notamment sa collecte, son stockage et son interrogation.

Ce travail de thèse traite les problématiques de modélisation et de gestion de provenance. Plus précisément, nous traitons les questions relatives à la modélisation de provenance et la conception de systèmes passant à l'échelle (ou scalables) assurant sa gestion appelés PMSs (de l'anglais, Provenance Management System). Ainsi, notre problématique de recherche est formulée par un ensemble de questions que nous traitons tout au long de ce travail dans une démarche globale. Nous répondons à chacune d'entre elles par une ou plusieurs propositions que nous validerons. Les questions qui résument notre problématique de recherche sont les suivantes :

1. ***Comment assurer l'interopérabilité syntaxique et sémantique entre des modèles de provenance hétérogènes issus de plusieurs systèmes et sources ?***
2. ***Comment modéliser la provenance et y intégrer des concepts et des propriétés métier ?***

3. ***Quelle est l'architecture de PMS la mieux adaptée pour concevoir un système couvrant la collecte, le stockage, la corrélation et l'interrogation de la provenance et quels sont ses modules ?***

4. ***Comment concevoir un PMS scalable qui permet de maximiser l'autonomie de sources de provenance distribuées ?***

Nous classons les questions relatives à notre problématique de recherche en deux catégories : une première catégorie focalisée sur **la modélisation de la provenance**. Elle traite les questions d'hétérogénéité syntaxique et sémantique entre modèles et données de provenance hétérogènes issus de plusieurs sources. Cette catégorie s'intéresse aussi à l'intégration des concepts et des propriétés métier dans les modèles et les requêtes de provenance. La deuxième catégorie est relative à **la conception de PMSs scalables**. Elle traite les questions de conception de PMSs, leurs architectures, leurs modules et leurs fonctionnalités. Elle s'intéresse aussi aux questions relatives au traitement de requêtes de provenance dans un environnement distribué ainsi qu'à la scalabilité de ces architectures. Nous déclinons à partir de ces problématiques les exigences sur le modèle de provenance et l'architecture de PMS que nous voulons concevoir.

### 1.2.1 Exigences liées à la modélisation de la provenance

#### 1.2.1.1 Hétérogénéité syntaxique

Les sources de provenance sont hétérogènes, multiples et non standardisées. Elles produisent des données de provenance ayant des syntaxes hétérogènes. On trouve des applications qui produisent des données de provenance dans un format texte par exemple, d'autres applications sont développées pour produire des traces avec une structure spécifique sur les logs des serveurs qui les hébergent et d'autres définissent leur propre syntaxe pour la génération de la provenance selon un modèle relationnel. Un problème d'interopérabilité syntaxique apparaît, l'analyse et le corrélation des données à travers ces différentes sources devient d'autant plus complexe que le nombre de sources et leur hétérogénéité augmente.

#### 1.2.1.2 Hétérogénéité sémantique

La provenance peut avoir une syntaxe unifiée basée sur un format et une structure pivot. Le problème est qu'une hétérogénéité sémantique peut apparaître entre des données syntaxiquement interopérables. En effet, des descriptions de provenance structurées dans une même syntaxe peuvent avoir des termes identiques ayant une sémantique différente. Inversement, des termes différents de cette syntaxe peuvent désigner le même objet ou le même élément. Il s'agit d'une problématique d'interopérabilité sémantique entre les modèles de provenance que notre approche de modélisation doit traiter.

#### 1.2.1.3 Couplage entre le document et sa provenance

Cette propriété implique qu'un document doit être lié à sa provenance. Effectivement, la provenance doit décrire de façon précise et complète l'état courant d'un document. Si un document A dépend d'un document B alors B doit apparaître dans la provenance de A. Aussi, si B apparaît dans la provenance de A, alors A doit dépendre de B. Cette propriété est très importante surtout que les données de provenance sont utilisées pour assurer la valeur probante des documents. Sans cette propriété, nous risquons d'utiliser des données de provenance qui ne sont pas cohérentes avec l'état du document ou d'utiliser des informations d'un document qui ne correspondent à sa provenance.

### 1.2.2 Modélisation des concepts et des propriétés métier

Chaque application possède des concepts et des propriétés métier qui lui sont spécifiques. Nous avons besoin de pouvoir intégrer ces concepts et propriétés dans des modèles de provenance. Cependant, nous devons garder notre modèle générique pour être indépendant des applications et donc éviter de concevoir des modèles spécifiques et non interopérables.

### 1.2.3 Exigences liées à la conception de PMSs

#### 1.2.3.1 Collecte de provenance

Les sources et les systèmes de provenance sont multiples, distribuées et autonomes. Idéalement, ces systèmes doivent fournir la provenance de façon automatique préservant la distribution des systèmes sources. Il est essentiel de définir un mécanisme de collecte qui soit le moins intrusif et qui n'impose pas des modifications ou des intégrations complexes du coté des systèmes sources.

#### 1.2.3.2 Identification des objets distribués

Dans un système centralisé où tous les objets numériques ainsi que leurs identifiants sont gérés de bout en bout, les problèmes d'identification ne se posent pas. Dans un contexte distribué, les objets numériques sont échangés et partagés entre plusieurs systèmes. Chacun de ces systèmes identifie ses objets avec un système d'identification qui lui est propre. Ainsi, un même objet numérique peut avoir plusieurs identifiants. Le croisement de ces identifiants est souvent problématique et la chaine de provenance d'un objet numérique devient inconnue si nous n'arrivons pas à connaitre l'ensemble de ces identifiants.

#### 1.2.3.3 Système de requêtes expressif avec plusieurs modèles d'interrogation

Un système de gestion de provenance doit permettre d'exprimer des requêtes et de les traiter sur les données de provenance qu'il sauvegarde. Ces requêtes peuvent varier des plus basiques au plus complexes. Elles peuvent être à forte sémantique et intégrer des concepts et des propriétés métier spécifiques.

#### 1.2.3.4 Passage à l'échelle

Les sources de provenance sont multiples. Comme il s'agit de données qui sont en mode d'ajout seulement, nous nous retrouvons souvent avec des volumes de provenance très importants. L'architecture de PMS proposée doit traiter et assurer le passage à l'échelle tant au niveau du stockage et que de l'exploitation (interrogation).

## 1.3 Un aperçu sur la thèse

### 1.3.1 Objectifs et contributions

#### 1.3.1.1 Objectifs de la thèse

Notre objectif est de proposer une approche de modélisation de provenance et une architecture de PMS permettant de :

- prendre en compte l'hétérogénéité syntaxique et sémantique des données et des modèles de provenance,
- modéliser et intégrer les aspects métier dans la provenance et pouvoir exprimer des requêtes à forte valeur métier,
- palier aux problématiques d'identifiants multiples entre les objets distribués pour pouvoir effectuer des opérations de corrélation entre des sources de provenance distribuées,
- permettre plusieurs interprétations métier des mêmes données de provenance,
- concevoir des architectures de PMSs assurant le passage à l'échelle au niveau du stockage de la provenance et son exploitation (interrogation).

### 1.3.2 Contributions

Les contributions de notre thèse sont :

1. La proposition d'une approche de modélisation de provenance. Elle se base sur un modèle de provenance minimal, indépendant des spécificités métier et qui sert comme modèle commun et partagé entre différentes sources de provenance hétérogènes et distribuées. Ce modèle minimal peut être enrichi avec des concepts et des propriétés métiers. Il est basé sur un standard de provenance, OPM [3] qui est caractérisé par sa généricité et son indépendance des technologies d'implémentation. Ce modèle nous permet d'assurer une interopérabilité syntaxique entre les différentes sources de provenance.
2. L'utilisation des technologies du Web sémantique pour assurer l'interopérabilité entre les modèles de provenance et pour faciliter la modélisation des concepts et des propriétés métier. La modélisation, l'encapsulation de propriétés métier et la

3. Open Provenance Model, www.openprovenance.org

définition des relations d'héritage entre les modèles est facilitée par l'utilisation des ontologies.

3. La proposition d'un mécanisme de généralisation des modèles de provenance instanciés pour générer des vues métier permettant ainsi d'avoir plusieurs visions métier sur les mêmes données de provenance.
4. La proposition de mécanismes de fusion et de corrélation permettant de lier la provenance de documents gérée dans différentes sources.
5. La proposition d'une architecture logique pour un PMS mettant en œvre notre approche de modélisation et proposant toutes les fonctionnalités nécessaires pour la gestion des modèles et des données de provenance.
6. La proposition d'une architecture de PMS dans un contexte distribué préservant la distribution des sources. Cette version distribuée se base sur un médiateur et propose plusieurs optimisations pour le traitement des requêtes de provenance permettant d'obtenir des réponses pertinentes à moindre coût.
7. L'implémentation de deux instances de PMS, basées sur notre architecture logique, l'une utilisant un RDF store et l'autre un SGBD NoSQL. Des mesures de performance effectuées sur des données réelles issues de Novapost nous ont permis de valider nos propositions.

Bien que nos objectifs et les axes de travail que nous avons identifiés sont génériques et s'intéressent à la provenance des données numériques, nous soulignons que nos modèles, propositions et expérimentations ont été réalisé uniquement sur les documents électroniques.

## 1.4 Organisation de la thèse

Ce mémoire de thèse se compose d'une introduction générale, de quatre chapitres et d'une conclusion générale. Le chapitre 2 propose un état de l'art sur les travaux dans le domaine de la modélisation de la provenance et de la conception de PMS. Nous présentons au début de ce chapitre quelques notions terminologiques relatives à la provenance, ses systèmes, ses modèles et ses modes de collecte. Ensuite, nous présentons une étude bibliographique sur les travaux de modélisation de provenance et les architectures de PMSs. Ainsi la première partie de cette étude s'intéresse aux modèles de provenance alors que la deuxième s'intéresse aux travaux liés à la conception d'architectures de PMSs ou à leurs implémentations. Enfin, nous étudions les limites des travaux présentés et en quoi ils ne correspondent pas à nos exigences annoncées au départ.

Les chapitres 3, 4 et 5 constituent le cœur de notre travail. Nous détaillons dans le chapitre 3 notre approche pour la définition d 'un cadre sémantique pour la modélisation et la gestion de provenance. Le chapitre 4 est dédié à la présentation de nos propositions d'architectures de PMS. Finalement, nous validons dans le chapitre 5 l'ensemble de nos propositions d'un point de vue fonctionnel et non-fonctionnel.

Dans le chapitre 3, nous présentons notre approche pour la modélisation et la gestion de la provenance. Elle se base sur l'utilisation des technologies du Web sémantique : RDF pour la description et le stockage des données de provenance, les ontologies pour la modélisation des concepts et des propriétés métier et SPARQL pour leur interrogation. Notre approche se base sur un modèle minimal (qui s'inspire du modèle OPM) assurant l'interopérabilité syntaxique entre les modèles de provenance correspondants à plusieurs sources hétérogènes ainsi que sur la notion de modèle de domaine qui est une spécialisation du modèle minimal. Ces modèles de domaine regroupent les concepts et les propriétés d'un métier donné et peuvent être instanciés pour créer des modèles de domaine instanciés. Nous introduisons un processus pour la généralisation de ces modèles instanciés vers des modèles plus abstraits ce qui permet de créer des vues métier différentes sur les mêmes données de provenance.

Dans le chapitre 4, nous nous focalisons sur la conception des systèmes de gestion de provenance. Nous présentons l'architecture logique d'un PMS qui se base sur le modèle de provenance que nous avons proposé dans le chapitre 3. Cette architecture logique est composée d'un module de traitement de requêtes, d'un module d'import de provenance, d'un module pour la gestion des modèles de domaine instanciés, d'un module de gestion des modèles de domaine, d'un module pour la fusion des modèles de domaine instanciés et d'un module de stockage. Nous détaillons un processus de fusion permettant d'unifier plusieurs modèles instanciés issus de plusieurs sources de provenance en un modèle instancié unique. Ce processus de fusion permet d'introduire de nouvelles connaissances et d'assurer la complétude des chaines de provenance. Nous introduisons aussi une technique de corrélation entre les objets distribués lors du traitement de requêtes. Cette technique est similaire à la fusion sauf qu'elle est effectuée à la demande et n'augmente pas la taille des données stockées. Ensuite, nous proposons une architecture de PMS dans un contexte distribué maximisant l'autonomie des sources. Elle se base sur un médiateur sur lequel nous implémentons un algorithme de traitement et de distribution des requêtes. Cet algorithme propose des optimisations pour le traitement des requêtes permettant d'obtenir des réponses pertinentes à moindre coût.

Dans le chapitre 5, nous nous intéressons à la validation de l'ensemble de nos propositions dans un contexte industriel relatif à la gestion du cycle de vie des documents électroniques chez Novapost. Nous commençons par la présentation d'une validation fonctionnelle de notre approche de modélisation sur un scénario de facturation dans un environnement Cloud faisant intervenir un Cloud broker. Cette validation fonctionnelle vise à montrer que notre approche de modélisation est générique, extensible et peut être exploitée dans des contextes autres que ceux de la gestion de documents électroniques.

Nous présentons ensuite les développement effectués chez Novapost pour la mise en place de services de traçabilité générant des données de provenance en XML. Ensuite, nous proposons une solution pour l'interfaçage de ces services dans notre architecture de PMS.

Ensuite, nous testons l'aspect passage à l'échelle de notre architecture. Ce test est réalisé par l'implémentation de deux prototypes de PMS sur deux technologies de stockage

différentes : un système RDF (Sesame) et un SGBD NoSQL (CouchDB). Les tests de montée en charge effectués sur les données de provenance Novapost ont montré les limites de Sesame tant pour le stockage que pour l'interrogation en utilisant le langage de requêtes SPARQL, alors que la version CouchDB associée à un langage de requêtes basé sur map/reduce a démontré sa capacité à suivre la charge de manière linéaire en augmentant le nombre de serveurs. Finalement, le chapitre 6 résume nos contributions et présente quelques perspectives de nos travaux de recherche.

*Chapitre 2*

# Contexte et État de l'art

*"S'il m'a été donné de voir un peu plus loin que les autres, c'est parce que j'étais monté sur les épaules de géants."*

Isaac Newton

## Sommaire

## 2.1 Introduction

Ce chapitre présente un travail de recherche bibliographique et est composé de deux parties. La première partie introduit un ensemble de notions terminologiques et de définitions nécessaires pour la compréhension de notre thèse. Elle définit le terme provenance, sa connotation précise dans notre travail et ses différents cas d'utilisation. Elle introduit ensuite les modes de collecte de la provenance, les modèles de provenance et les systèmes de gestion de provenance.

La deuxième partie présente un état de l'art sur les travaux menés par la communauté de provenance et qui s'intéressent à la modélisation, à la collecte, à l'intégration et l'interrogation de la provenance. On clôture ce chapitre par une synthèse sur ces différents travaux récapitulant les caractéristiques et les limites des différents modèles et systèmes présentés.

## 2.2 Notions terminologiques et cas d'utilisation de la provenance

### 2.2.1 Notions terminologiques

#### 2.2.1.1 Définitions

Selon le dictionnaire Larousse, la provenance est l'endroit d'où quelque chose vient ou provient. C'est le fait de tirer son origine de quelqu'un, de quelque chose, l'avoir comme source, en être extrait. Ainsi, la provenance peut être considérée comme les étapes décrivant la dérivation d'une source vers un état spécifique. Cette description peut prendre différentes formes et contenir des propriétés différentes selon l'intérêt et l'usage prévu. Pour une œuvre artistique par exemple (sculpture, tableau...), la provenance doit identifier son réalisateur, la chaîne des propriétaires, les restaurations et les modifications effectuées, leurs dates ainsi que les entités qui les ont effectué. Cette définition identifie deux compréhensions du terme provenance : d'une part comme un concept désignant la source ou la dérivation d'un objet, d'autre part comme une référence sur l'enregistrement d'une telle dérivation.

Dans un cadre de gestion de qualité et selon le standard ISO 9000 2005, la provenance est liée à l'aptitude de retrouver l'historique, l'utilisation ou la localisation d'une entité (activité, processus, produit...) au moyen d'identifications enregistrées.

Dans le domaine de la gestion électronique de documents, cette notion de provenance est très importante. En effet, les différents standards de gestion et d'archivage de documents électroniques s'y intéressent et exigent de l'enregistrer pour pouvoir l'utiliser en cas d'audit. L' ISO 15489-1 [ISO15489-1, 2001] standard de gestion d'archives définit la gestion de provenance comme étant *le fait de créer, enregistrer et préserver les données relatives aux mouvements et à l'utilisation des documents.*

#### 2.2.1.2 Utilisations du terme provenance

Le mot provenance tout court est vague et peut avoir plusieurs significations qui varient selon le domaine d'application et le contexte d'utilisation. Dans un travail de positionnement, Moreau [Moreau, 2006] définit une carte conceptuelle pour l'utilisation du terme provenance (voir Figure 2.1). Son travail identifie trois types d'utilisation de ce terme :

- quelque chose de provenance : plusieurs termes comme *enregistrement de provenance*, *traces de provenance* existent et se réfèrent généralement à une représentation informatique de la provenance. Dans cette famille, nous trouvons aussi *architecture de provenance*, *système de gestion de provenance* et *modèle de provenance* qui se réfèrent généralement à une infrastructure logicielle capable de capturer cette information et de l'utiliser.
- provenance de quelque chose : *provenance d'une donnée*, *provenance d'un document*. Cette utilisation désigne l'origine de la donnée ou du document. Elle peut être remplacée par le mot **provenance** tout court.
- faire quelque chose pour/avec/sur : comme *tracer la provenance*, *capturer la provenance*, *interroger la provenance*. Cette utilisation ne fait pas une distinction entre la provenance comme concept et la provenance comme représentation.

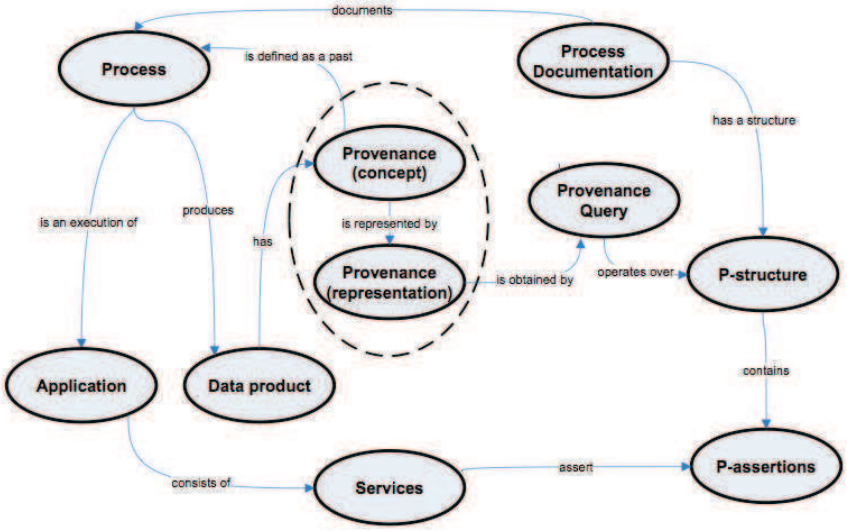


FIGURE 2.1 – Carte conceptuelle de la provenance [Moreau, 2006]

#### 2.2.1.3 Modèle de provenance

Un modèle de provenance est une structure qui décrit l'information de provenance à retenir. Idéalement, ce modèle doit permettre de décrire avec précision toutes les informations sur les actions, les transformations effectuées sur une donnée ainsi que les acteurs qui

les ont réalisés. Dans ce cas, il devient possible que les instances du modèle permettent de reproduire des processus et des scénarios d'exécution et d'obtenir les mêmes résultats que les exécutions précédentes.

On distingue principalement les deux types de modèle de provenance suivants :

- Modèle de provenance abstrait : ce modèle est générique, il ne contient pas les informations spécifiques à un métier précis. Il décrit les informations minimales de provenance qui doivent exister dans des contextes métier différents. Il contient par exemple des informations sur l'identifiant d' un objet, la date de réalisation d'une action et l'identifiant de l'entité l'effectuant. Ayant une structure minimale faisant abstraction des détails métier, ce modèle est interopérable. Nous définissons cette propriété d'interopérabilité entre modèles comme leur capacité à modéliser et interpréter des données et de pouvoir les utiliser sans aucune ambiguïté.
- Modèle de provenance métier : c'est un modèle spécialisé qui décrit la provenance pour un un métier spécifique. Il est sémantiquement plus riche que le modèle abstrait. Par contre il n'est pas interopérable ni réutilisable hors du métier pour lequel il a été défini parce qu'il contient des concepts et des propriétés qui lui sont spécifiques.

#### 2.2.1.4 Système de gestion de provenance

Un système de gestion de provenance (PMS) est un système informatique qui collecte la provenance à partir d'une ou de plusieurs sources (une application Web, un système d'information, une base de données, un système de fichiers...) et qui définit une politique pour sa gestion (sauvegarde, accès et interrogation). Ainsi, un tel système est composé de trois composants principaux qui sont le composant de capture (ou de collecte), le composant de stockage et le composant d'accès et d'interrogation.

#### 2.2.1.5 Modes de collecte de la provenance

Un PMS doit définir un mode de collecte lui permettant de capturer la provenance. Deux modes de collecte de provenance existent :

- **Mode observé :** le PMS observe tous les événements qui se produisent et infère de façon transparente la provenance. Dans ce type de configuration, le PMS ne nécessite pas une intervention pour modifier le système source. Son comportement par défaut est de collecter et de sauvegarder la provenance de toutes les données. L'inconvénient de ce mode est que la provenance collectée n'est pas sémantiquement riche. C'est le cas par exemple pour les serveurs Web, les bases de données et les systèmes d'exploitation qui génèrent de façon automatique la provenance dans des fichiers de logs.
- **Mode assisté :** le PMS se base dans ce cas sur des sources de provenance personnalisées. Les systèmes sources de provenance doivent être modifiés pour générer la provenance selon un modèle métier qui leur est spécifique. Dans ce mode, la

provenance est plus riche sémantiquement que le mode observé parce qu'elle est personnalisée avec un modèle métier. Par contre, ce mode est intrusif parce qu'il nécessite un effort d'intégration du coté des systèmes sources ce qui requiert de leur apporter des modifications.

Chacun des deux modes possède ses avantages et ses inconvénients. Cependant, ils sont complémentaires : un système de provenance idéal fournit une provenance sémantiquement riche contenant tous les détails métiers d'une application en se basant sur une collecte transparente et automatique. Ceci est possible en utilisant le mode observé par une application qui se base sur la provenance assistée d'une autre application.

### 2.2.2 Cas d'utilisation de la provenance

Il y a peu, la provenance n'était pas considérée comme une information très importante ou très utile. Aussi, sa gestion était considérée comme une tâche lourde qui engendre des volumes de données additionnels, qui nécessite une politique de gestion, de stockage et d'accès dédiée sans avoir une vraie valeur ajoutée.

Aujourd'hui, avec l'évolution des architectures distribuées et orientées service, l'émergence du Cloud computing, la dématérialisation croissante et donc la production exponentielle d'informations, la provenance devient de plus en plus importante, voir critique. Cette information peut être utilisée pour répondre à des questions critiques dans des cas d'utilisations différents. Des travaux de recherche récents la considèrent comme une donnée de premier ordre [Muniswamy-Reddy et Seltzer, 2010] et d'autres la classifient parmi les grands challenges de l'informatique du futur [UKCRC, 2008]. Parmi les cas d'utilisation les plus importants, nous citons :

- **Qualité de données :** connaissant la provenance d'une donnée, il est possible de juger sa qualité, de savoir si elle satisfait les exigences techniques, métier et légales d'un métier donné.
- **Reproduction de processus :** la provenance n'est pas seulement une trace de ce qui a été fait mais aussi un moyen de le reproduire. Ce cas d'utilisation est courant et très utile pour les expériences scientifiques par exemple (en biologie, sciences de la terre ...) et pour la communauté d'E-science. Dans une telle utilisation, la complétude de la provenance ainsi que sa précision sont cruciales. Pour une expérience scientifique par exemple, tous les détails sur les conditions de son déroulement, les instruments utilisés, les logiciels utilisés ainsi que leurs versions doivent être enregistrés pour pouvoir reproduire l'expérience et aboutir aux mêmes résultats que l'expérience initiale.
- **Sécurité et détection d'intrusion :** la provenance est très utile pour détecter les attaques sur les applications. Elle permet de détecter à quel niveau l'attaque a été effectuée et d'identifier sa cause principale. Elle permet aussi de prévoir si l'attaque originale va lancer ou propager de nouvelles attaques sur d'autres couches de l'application.

– **Conformité légale et audit :** un historique précis et certifié sur les traitements effectués sur des données peut servir comme preuve de conformité aux normes et aux réglementations en vigueur. En cas de conflit entre deux entités sur l'authenticité d'une donnée, la provenance peut servir comme un élément de preuve légale.
– **Gestion du cycle de vie de l'information :** Généralement, la valeur métier d'une donnée n'est pas connue parce que l'association entre l'information et la donnée n'est pas clairement établie ou n'existe pas. La gestion du cycle de vie de l'information ou ILM [SNIA, 2004] (Information Lifecycle Management) comprend des politiques, des processus, des pratiques et des outils permettant d'aligner la valeur métier de l'information avec les données depuis leurs création et jusqu'à leurs suppression. Les politiques ILM se basent souvent sur des règles qui ne sont pas automatisées mais plutôt définies manuellement. Pour cette raison, l'utilisation de la provenance permet de réduire cet écart entre le métier et la technique en fournissant une source supplémentaire qui relie la donnée à son contexte métier d'utilisation.

## 2.3 Travaux liés à la modélisation de la provenance

Dans cette partie, nous nous intéressons aux différents travaux qui ont traité la provenance et sa modélisation.

Buneman et al. [Buneman *et al.*, 2000] se sont intéressés à l'annotation, l'archivage et le versionning des bases de données scientifiques. Ces bases de données sont souvent organisées manuellement en se basant sur des annotations. En effet, dans des sciences comme la biologie et l'astronomie, plusieurs bases de données servent aux scientifiques pour effectuer des expériences et générer de nouvelles données. Dans ce contexte, la provenance est exigée pour connaitre l'origine des données générées. Pour les bases qui changent au fil du temps, des techniques de versionning compact ont été introduites pour identifier la provenance des tuples. Ces techniques visent à identifier la base de donnée source ainsi que la version associée. Dans ces travaux, Buneman a mis l'accent sur deux questions pour caractériser la provenance : « le pourquoi » de la provenance (why) et son « où » (where).

Le pourquoi s'intéresse à l'existence d'une donnée dans la sortie d'une requête (ou d'une vue) et identifie les sources (tables, bases de données) qui justifient son existence. La réponse à cette question est un arbre référençant les sources responsables de l'existence de cette donnée. La deuxième question identifie les données qui ont permis de créer cette sortie. Buneman et al. [Buneman *et al.*, 2001] ont introduit deux algorithmes permettant de répondre à ces deux questions pour des bases de données relationnelles, orientées objet et semi-structurées où les requêtes satisfont un modèle déterministe. Ces deux algorithmes se basent sur l'inversion de requêtes pour déterminer la provenance des tuples. Les requêtes inverses sont associées aux données dérivées ce qui rend le modèle centré sur les données avec une granularité au niveau des tuples de la base.

Ram et Liu [Ram et Liu, 2009] jugent que le modèle de provenance présenté par

Buneman, basé uniquement sur le « pourquoi » et le « où » n'est pas très expressif. Ils proposent comme alternative un modèle ontologique plus riche pour exprimer la sémantique de la provenance. Ce modèle, appelé W7 a pour objectif de proposer un consensus sur la provenance entre les différentes communautés. Il est définit par sept questions dont la réponse permet de décrire de manière précise la provenance de n'importe quel type de donnée. Les questions définissant ce modèle sont : quoi (what), quand (when), où (where), comment (how), qui (who), quel (which) et pourquoi (why). Ainsi, la provenance d'une donnée D est un 7-uplet, et nous avons p(D) = <Quoi, Quand, Où, Comment, Qui, Quel, Pourquoi>(D).

- le « quoi » identifie un événement ou un changement appliqué à la donnée,
- le « quand » est relatif au temps de réalisation ou à la durée d'un événement,
- le « où » est le cadre spatial indiquant la localisation de déroulement d'un événement,
- le « comment » est l'action (ou la série d'actions) qui a déclenché un ou plusieurs événements. Un événement peut se produire quand il est initié par un acteur qui peut être un acteur humain ou logiciel,
- le « qui » se réfère à l'identité des acteurs ayant déclenché ou qui sont impliqués dans un événement,
- le « quel » se réfère aux instruments et/ou aux logiciels utilisés dans un événement,
- le « pourquoi » est l'ensemble d'éléments qui expliquent la cause de déroulement d'un événement.

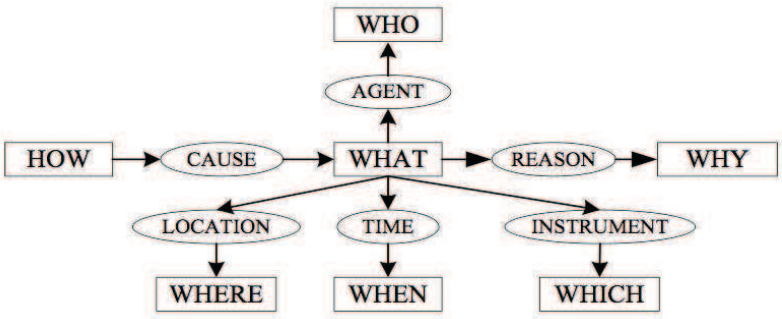


FIGURE 2.2 – Un Aperçu du modèle W7

L'ontologie W7 construite sur la base de ces questions est générique et indépendante du domaine. Cette généricité lui permet de couvrir des besoins de domaines différents. Cependant, et afin d'être d'intérêt pratique, elle doit être extensible. C'est pour cette raison que W7 utilise le mécanisme de définition de type développé par Sowa [Sowa, 1984] qui a été largement utilisé comme langage d'ontologie et qui permet entre autre de définir des extensions de domaine. W7 peut être considéré comme un modèle de provenance générique permettant d'encapsuler dans un même modèle ce qui est relatif à la provenance avec ce qui est relatif au métier. Il peut être mis en place avec un mode de collecte assisté pour répondre à toutes les questions du modèle. Par contre, W7 n'a été utilisé que dans des

scénarios de validation fonctionnelle et non par la mise en place de systèmes de gestion de provenance industriels.

Pour fédérer les efforts autour de la modélisation de la provenance et afin de proposer un modèle générique, indépendant des technologies et des domaines d'application, la communauté des workflows scientifiques a proposé OPM [1] [Moreau *et al.*, 2010a]. OPM est un modèle de graphe générique pour la modélisation de la provenance. Il a été conçu à la base pour assurer l'interopérabilité de la provenance entre les systèmes hétérogènes. Il se base sur trois entités principales qui sont ***Artifact***, ***Process*** et ***Agent***. Cinq relations de causalité relient ces entités (voir Figure 2.3). Ces relations peuvent être annotées pour permettre une meilleure compréhension du graphe de provenance. Les graphes OPM peuvent être sémantiquement enrichis en utilisant la notion de profil (Profile) qui permet de décrire un ensemble de spécificités d'un métier.

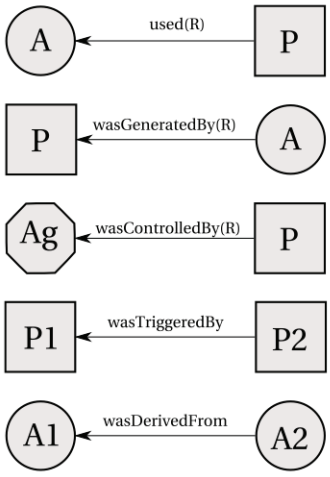


FIGURE 2.3 – Les relations de causalité dans le modèle OPM : les sources sont les effets, les destinations sont les causes

L'avantage du modèle OPM est sa structure générique et extensible. En plus, il possède un aspect abstrait, indépendant de toute technologie d'implémentation ou format de stockage. Rajoutons à cela que la communauté qui contribue, développe et utilise OPM a bien évolué pendant les dernières années. Cette évolution peut être illustré par le nombre d'outils développés autour d'OPM comme la boite à outils OPM (OPM toolbox [2]), Tupelo [3], ProvenanceJS [Groth, 2010], Karma [Cao *et al.*, 2009], ourSpaces [4] ainsi qu'aux différents systèmes qui proposent des fonctions d'export de provenance compatibles avec OPM comme

1. Open Provenance Model : www.openprovenance.org
2. www.openprovenance.org/toolbox.html
3. www.tupeloproject.ncsa.uiuc.edu
4. www.policygrid.org

Taverna [5], VisTrails [6], Swift [7] ou Kepler [8]. Cependant, nous pouvons remarquer, malgré l'aspect générique d'OPM que tous ces outils sont spécifiques aux workflows scientifiques. Effectivement, OPM n'a été validé conceptuellement et formellement que dans ce cadre là. Par contre, des scénarios distribués, mettant en œuvre plusieurs types de systèmes et de sources de provenance autres que les systèmes de workflow est un travail en cours. Aussi, plusieurs questions se posent aujourd'hui en ce qui concerne l'utilisation des profils et les règles d'inférence dans OPM. Les profils assurent l'enrichissement sémantique, cependant les règles d'inférence peuvent devenir contradictoires avec le modèle de base ou introduire un manque de précision [Simmhan, 2009].

### 2.3.1 Synthèse

A travers cette étude bibliographique sur les modèles de provenance, nous avons pu identifier deux catégories de modèles : une première catégorie qui présente un modèle riche contenant ce qui est spécifique à la provenance et aux spécificités métier (comme le modèle W7 [Ram et Liu, 2009]). La deuxième catégorie se base sur l'utilisation d'un modèle de provenance minimal, indépendant du contexte d'utilisation et qui n'est pas très riche en terme sémantique. Cependant, ce modèle est extensible et peut être enrichi avec des descriptions métier (comme le modèle OPM [Moreau *et al.*, 2010a] qui peut être considéré comme référence pour l'interopérabilité entre modèles de provenance). La première catégorie de modèles n'est pas flexible. Elle impose l'utilisation d'un modèle unique pour toutes les sources de provenance (pour le stockage et l'interrogation). Par contre, son avantage est la facilité d'interrogation : les sources ne peuvent être interrogée qu'avec ce modèle unique. L'inconvénient de ce type de modèle est que sa modification implique une modification sur toutes les sources qui l'utilisent. Par contre, l'utilisation d'un modèle de provenance extensible permet de garder les sources autonomes : une changement du modèle de provenance au niveau d'une source ne nécessite pas sa propagation sur les autres source. Par contre, son inconvénient est que l'interrogation devient plus difficile puisque plusieurs modèles d'interrogation existent.

Cette étude bibliographique sur les modèles de provenance nous a permis d'identifier un ensemble d'exigences. Un modèle de provenance doit assurer :

1. la généricité et l'indépendance des systèmes sources,
2. l'extensibilité,
3. l'interopérabilité syntaxique,
4. l'interopérabilité sémantique,
5. la fusion et la corrélation entre sources de provenance,

5. www.taverna.org.uk
6. www.vistrails.org
7. www.ci.uchicago.edu/swift
8. www.kepler-project.org

6. la création des vues métiers.

Ces exigences sont décrites dans le tableau suivant (voir Table 2.1) qui illustre pour chaque exigence le ou les modèles de l'état de l'art qui la prennent en compte.

| Type d'exigence | Description | Exemple de modèle |
|---|---|---|
| Généricité et indépendance des systèmes sources | Le modèle ne possède pas des concepts liés à un métier spécifique. Il peut servir comme base de modèle de provenance pour plusieurs types de systèmes sources. | OPM, W7 |
| Extensibilité | Le modèle peut être étendu pour définir de nouveaux modèles. | OPM (avec les profils) |
| Interopérabilité syntaxique | Avoir une structure qui sert comme base minimale pour la description de la provenance pour différents systèmes sources. Cette structure doit avoir plusieurs sérialisations vers différentes syntaxes. | OPM |
| Interopérabilité sémantique | Mettre en place des concepts et des propriétés compréhensibles et ayant le même sens pour les différents systèmes sources. | |
| Fusion entre sources de provenance | Permettre de fusionner des modèles de provenance et leurs instances selon un seul modèle. La compatibilité entre les modèles à fusionner doit être assurée. | |
| Corrélation entre sources de provenance | Détecter des liens de corrélation qui existent entres des artifacts distribués. Cette corrélation sert pour assurer la complétude des chaines de provenance. | |
| Création des vues métier | Permettre de donner plusieurs visions sur les mêmes données de provenance correspondant à différents modèles métier. | |

TABLE 2.1: Exigences sur les modèles de provenance

## 2.4 Travaux liés aux systèmes de gestion de provenance

Dans cette partie, nous nous intéressons aux travaux qui ont proposé des architectures et/ou des implémentation de systèmes de gestion de provenance.

### 2.4.1 Provenance dans le domaine des bases de données

Dans ce domaine, identifier l'ensemble maximal de tuples ayant produit une donnée dans une vue matérialisée est une problématique de traçabilité assez connue. Widom [Widom, 2005] a proposé Trio (voir Figure 2.4) qui est un système de gestion de base de données appartenant à la famille ULDBs (Uncertainty and Lineage Databases). Ce type de système intègre nativement les dimensions provenance (lineage) et incertitude (uncertainty) comme des données de premier ordre. Trio est conçu en ajoutant une nouvelle couche au modèle relationnel et une autre couche pour la réécriture des requêtes SQL. La motivation initiale pour le développement de Trio était la gestion de la provenance dans les entrepôts de données.

Trio se base sur une approche d'inversion qui lui permet d'identifier le nombre maximal de tuples qui ont produit un tuple dans un entrepôt. Cette approche considère que la provenance d'un tuple (ou d'un ensemble de tuples) dans une base de données peut être modélisée comme un arbre de requêtes qu'on peut évaluer selon une approche bottom-up en partant des opérateurs fils ayant les tables en entrée et les opérateurs parents successifs qui prennent en entrée les relations résultantes des opérateurs fils. Sur ce principe, et pour toute requête de type ASPJ (Aggregate-Select-Project-Join), il est possible de créer automatiquement la requête inverse. Les requêtes inverses sont enregistrées dans une table « Lineage » spéciale. Pour chaque tuple, sa date de création et son type de dérivation (Insert, Update, Vue ou procédure stockée) sont enregistrés. Cette table peut être interrogée avec TriQL (Trio Query Language) qui est un langage de requête spécifique à Trio. Cette dimension lineage dans Trio se résume dans l'identification des tables et des bases de données sources qui ont permis d'obtenir le tuple résultat. Cette association directe entre provenance et données rend la gestion de la provenance dans Trio centrée sur les données.

### 2.4.2 Provenance dans le domaine des workflows scientifiques et de l'E-Science

Les communautés de l'E-science et des workflows scientifiques ont accordé une attention particulière à la provenance. L'E-science est définie comme la science qui utilise d'immenses volumétries de données et qui requiert l'utilisation de technologies de collaboration distribuées comme les grilles de calcul. Ce domaine inclut les simulations sociales, la physique des particules, la biologie, la bio-informatique ... Dans ce domaine, le besoin d'enregistrer la provenance des calculs ou des expériences scientifiques réalisées est très important. Ceci pour des raisons de reproduction, d'audit ou encore de qualité de données. Dans ce contexte,

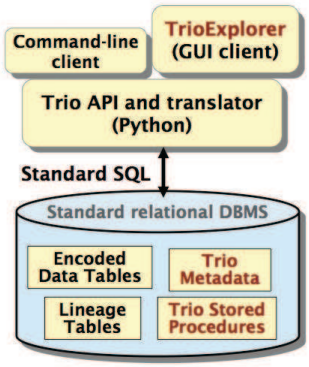


FIGURE 2.4 – Architecture du système Trio [Widom, 2005]

plusieurs travaux se sont intéressé à la génération de la provenance, à sa collecte et son exploitation.

Le projet myGrid [9] a été initié en 2003 et avait pour objectif de développer un middleware open source pour les expérimentations in silico dans le domaine de la biologie ainsi qu'une suite d'outils logiciels complémentaires pour aider les E-scientifiques à créer des laboratoires virtuels (E-laboratories) dans des domaines variés (biologie, science sociale, musique, astronomie, multimédia, chimie...) et de pouvoir les orchestrer. Ces expériences sont composées d'un ensemble de workflows qui s'exécutent sur des grilles de calcul [Stevens *et al.*, 2003]. Elles se basent sur l'analyse de données provenant de portails Web et sur un ensemble de procédures de calcul similaires à celles effectuées dans les laboratoires. Pour assurer la qualité et le fiabilité des résultats, une attention particulière a été accordée à la précision et l'exactitude de la provenance. Les contraintes de disponibilité des ressources de calcul et de performances viennent en second lieu. En exécutant les workflows écrits en XScufl (qui est la version XML du langage de workflow Scufl [10]) sur l'environnement Taverna [Zhao *et al.*, 2004], MyGrid génère et stocke des données de provenance. Ceci grâce à un mode de collecte assisté où les workflows et les services sont annotés avec des descriptions sémantiques. La provenance générée contient les services invoqués, leurs paramètres, les dates de début et de fin d'exécution, les données utilisées en entrée, les données dérivées et les résultats. Outre les méta-données contextuelles et organisationnelles telles que le propriétaire, le projet, les hypothèses et le contexte de l'expérience, les utilisateurs peuvent fournir des termes spécifiques au domaine qui permettront de mieux décrire l'expérience et ses données. Ceci facilite la compréhension des expériences effectuées et la validation de leurs résultats. La granularité des éléments pour lesquels la provenance est enregistrée est variable et configurable. Elle peut être l'expérience, le workflow, le service ou la donnée.

La multitude de systèmes et d'outils de gestion de provenance dans les environnements d'E-science et l'absence de standards normalisant une structure et un format d'échange de

9. www.mygrid.org.uk
10. Simple Conceptual Unified Flow Language

provenance ont rendu la collaboration entre eux impossible. Pour surmonter cette problématique, Groth et al. ont développé PASOA (Provenance Aware Service Oriented Architecture) qui est une infrastructure pour la collecte, le stockage, la gestion et l'interrogation de la provenance dans un environnement d'E-Science à base de services [Groth *et al.*, 2004]. PASOA se base sur un protocole ouvert (appelé PReP : Provenance Recording Protocol) qui vise à favoriser l'interopérabilité et l'échange de la provenance entre des acteurs utilisant des systèmes hétérogènes. Ce protocole définit 14 types de messages de provenance pour différents types d'interaction entre les acteurs et invocation de services. La granularité de la provenance possède le même niveau des entrées et des sorties du Web service. Tous les messages d'interaction et d'acteurs sont corrélés en utilisant un identifiant unique (appelé ActivityID) qui est échangé dans les messages de provenance.

En adoptant une architecture orientée service, plusieurs partenaires intervenant dans une expérience scientifique, utilisant des systèmes et des implémentations différentes peuvent échanger la provenance. PASOA définit aussi un ensemble d'exigences pour un système de gestion de provenance dans une architecture orientée service comme la vérifiabilité, l'attribution, la reproductibilité, la préservation à long terme, le passage à l'échelle, la généricité (pour pouvoir supporter plusieurs applications distribuées) et la personnalisation (pouvoir personnaliser la provenance collectée). Cependant, le mode de corrélation des identifiants d'acteurs avec les identifiants de processus rend PASOA centré uniquement sur les processus. PASOA nécessite aussi un effort d'intégration du coté des systèmes sources pour intégrer les appels au service de provenance avec les messages correspondants.

Dans le même contexte, Groth et al. ont proposé dans le cadre du projet européen *Provenance*[11] une architecture et une implémentation d'un PMS scalable pour les systèmes de grille [Groth *et al.*, 2006]. Cette architecture s'intéresse à la gestion de provenance d'un ensemble de processus, distribués à large échelle et s'exécutant dans un environnement de grille. Dans ce travail, les auteurs ont proposé les spécifications fonctionnelles détaillées pour un PMS ainsi que son architecture logique d'une manière indépendante des technologies d'implémentation et des infrastructures de déploiement.

Dans ce PMS, la provenance d'une donnée est représentée par un ensemble d'informations documentant le processus qui l'a généré. Dans ce travail, une structure appelée p-assertion (assertion de provenance) est définie. Il s'agit de la structure qui reflète une action, elle est réalisée par un acteur et appartient à un processus. Sur cette base, la provenance d'un processus est définie comme étant l'ensemble des p-assertions des acteurs impliqués dans ce processus. Cette architecture couvre le cycle de vie de la provenance à savoir :

- Création des p-assertions : au cours de l'exécution, les acteurs créent leurs p-assertions qui reflètent leur implication dans un (ou plusieurs) processus,
- Enregistrement : après la phase de création, les p-assertions sont enregistrées dans un entrepôt de provenance pour être réutilisées et permettre la reconstitution de la

11. http ://www.gridprovenance.org

provenance d'un processus,
- Interrogation : suite à la fin d'exécution d'un processus, les utilisateurs (ou les applications) peuvent interroger le dépôt de provenance,
- Gestion : le dépôt et son contenu peuvent être gérés pour intégrer les aspects de distribution, d'organisation, de changement de version, de migration ...

Pour les auteurs, tout PMS doit proposer des fonctions pour supporter ces quatre phases du cycle de vie de la provenance. Pour les assurer, cette architecture (voir Figure 2.5) propose principalement trois interfaces : une interface de collecte de provenance (librairie coté client pour la création et la sauvegarde des p-assertions), une interface de gestion (pour les opérations de gestion) et une interface de requêtes. L' avantage de cette architecture est qu'elle peut être appliquée dans un système de workflow ou dans une architecture orientée service. Ce travail est le premier à présenter une architecture logique de PMS détaillée, indépendante des technologies d'implémentation et qui couvre toutes les étapes de gestion du cycle de vie de la provenance. Cependant, cette architecture considère uniquement un mode de collecte assisté et est conçue dans une vision où tous les systèmes impliqués dans un processus sont connus et contrôlés. Elle est centrée sur les processus et a été expérimentée uniquement dans un environnement de workflow scientifique. Aussi, l'aspect d'exploitation et d'interrogation des p-assertions n'a pas été bien développé dans ce travail. En effet, il ne définit pas une interface d'accès et d'interrogation dans le protocole PReP et se limite à la proposition d'une API basique pour l'accès à la provenance collectée.

Dans [Chebotko *et al.*, 2010], les auteurs s'intéressent à l'exploitation de la provenance et son interrogation dans un environnement de workflows scientifiques. Les auteurs se sont focalisés sur cette problématique car ils considèrent que malgré les efforts de la communauté pour standardiser la modélisation de la provenance avec des modèles de référence comme OPM, il n'existe aujourd'hui aucun travail dédié aux problématiques d'interrogation de provenance. Étant donné que les systèmes de gestion de base de données relationnelles et le langage de requêtes SQL ont acquis une grande maturité pour le stockage et l'interrogation des données, ce travail propose d'utiliser ces technologies pour implémenter des PMSs. Il propose une approche de gestion de provenance hybride, exploitant d'une part les technologies du Web sémantique pour les besoins d'interopérabilité, d'extensibilité et d'inférence et celles des bases de données relationnelles et SQL pour le stockage et l'interrogation d'autre part.

En se basant sur cette approche, ce travail propose une architecture appelée *RDF_*PROV. Elle est composée de trois couches et se base sur une analyse des spécificités de la provenance dans les workflows scientifiques et une analyse des points faibles des RDF stores sémantiques testés. En effet, l'évaluation de deux RDF stores (Jena et Sesame) a montré des problèmes de performance pour le traitement des requêtes complexes ainsi que pour le passage à l'échelle sur de gros volumes de données. Dans cette architecture (voir Figure 2.6), la couche modèle de provenance est responsable de la gestion des ontologies et des règles d'inférence pour enrichir les données RDF à enregistrer. La deuxième couche est celle de mapping.

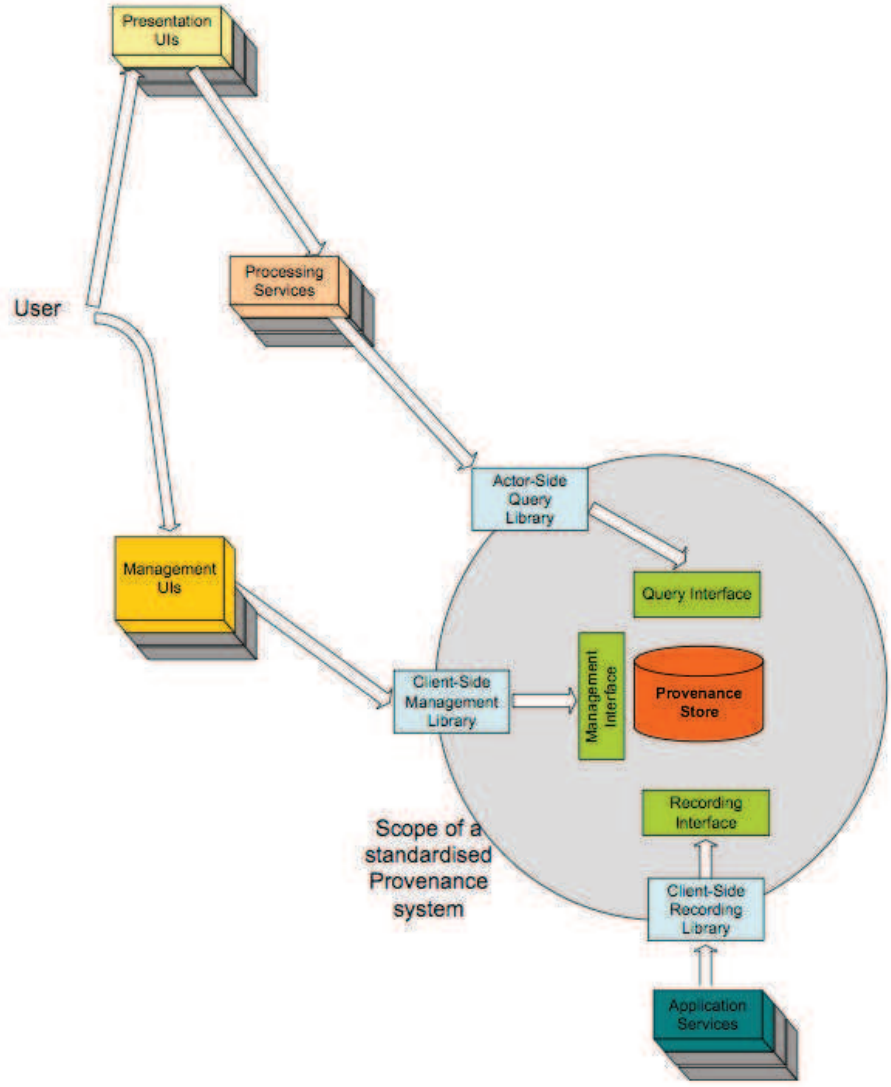


FIGURE 2.5 – Architecture logique de provenance proposée par Groth et al. [Groth *et al.*, 2006]

Elle se compose de trois mappings : (1) le mapping de schémas qui permet de générer un schéma relationnel à partir d'une ontologie de provenance, (2) le mapping de données qui fait la correspondance entre les triplets RDF et les tuples relationnels et (3) la mapping de requêtes qui traduit des requêtes sémantiques formalisées en SPARQL en des requêtes relationnelles formalisées en SQL. Ces mappings permettront de faire le lien entre le modèle de provenance sémantique et le modèle relationnel qui sera matérialisé dans un SGBD relationnel qui peut être interrogé en SQL. *RDF_*PROV a permis d'exprimer les différentes requêtes et les temps de traitement sont meilleurs que les temps de traitement sur les RDF stores testés. Cependant, la stratégie de mapping utilisée influence beaucoup les temps de traitement des requêtes. Un compromis entre le temps de réponse souhaité et le type du workflow (workflow de longue durée et workflow de courte durée) doit être effectué.

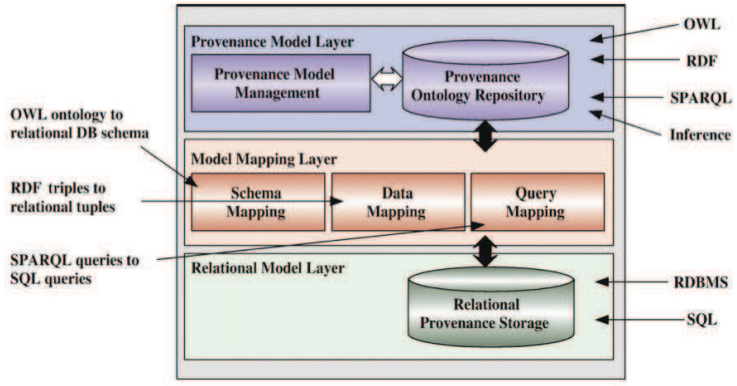


FIGURE 2.6 – Architecture de *RDF_PROV*

Les auteurs soulignent aussi que le schéma de la base de données est complexe et que son administration et sa migration sont des tâches complexes. Pour cette raison, ils proposent d'explorer les bases de données NoSQL orientées colonnes qui possèdent l'avantage d'être extensibles et sans schéma [Abadi *et al.*, 2007], [Sidirourgos *et al.*, 2008].

### 2.4.3 Provenance dans le contexte du Web et des données liées

Dans un autre contexte, celui du Web et des données liées, plusieurs travaux ce sont intéressés aux systèmes et aux applications qui consomment, échangent et générèrent des données de provenance sur le Web.

Ce terme, données liées (en anglais Linked Data) a été inventé par Tim Berners-Lee [Berners-Lee *et al.*, 2001]. Il se réfère à un style de publication et d'interconnexion des données structurées sur le Web. L'hypothèse de base est que plus on crée des liens entre les données, plus leur valeur et leur utilité augmente. Les données liées concernent donc l'utilisation du Web pour publier, partager des données structurées et automatiser leur interprétation et leur exploitation.

Les données liées se basent sur les principes du Web sémantique qui sont l'utilisation du modèle RDF pour publier les données structurées sur le Web et l'utilisation des liens RDF pour interconnecter les données provenant de sources différentes. Ce type de création, de publication et d'échange de données introduit des questions sur la fiabilité et l'intégrité des contenus et positionne la provenance comme une problématique centrale. Plusieurs travaux de recherche comme [Sahoo *et al.*, 2009], [Hartig, 2009] et [Freitas *et al.*, 2010] se sont intéresses à la provenance dans un contexte « Linked Data » et aux différentes questions de publication, consommation, gestion et interrogation de la provenance.

Cette vision du Web, appelée Web sémantique est un terme qui a été inventé par Tim Berners-Lee [Berners-Lee, 1998] pour introduire une nouvelle notion du Web où l'information est partagée, interprétée et exploitée pas uniquement par les humains mais aussi entre les machines. Le Web sémantique peut être vu comme l'enrichissement du

Web traditionnel avec de nouvelles techniques permettant aux machines d'exploiter les données, de les partager et de permettre ainsi aux utilisateurs de naviguer, combiner et exploiter l'information dans ce vaste répertoire de connaissances. Cette vision repose sur une architecture en couches (voir Figure 2.7).

Pour mieux comprendre les systèmes de gestion de provenance réalisés en se basant sur les technologies du Web sémantique, nous allons commencer par l'introduction de cette notion et de ses technologies. Selon Berners Lee [Berners-Lee *et al.*, 2001], le Web sémantique peut être vu comme l'empilement d'un ensemble de couches. Les couches les plus basses assurent l'interopérabilité syntaxique : la notion d'URI [12] assure un référencement standard et universel qui permet d'identifier les ressources. Unicode est l'encodage textuel universel utilisé pour échanger des symboles. Les URLs [13], comme les URIs sont des chaines de caractères utilisées pour identifier des ressources par leur localisation.

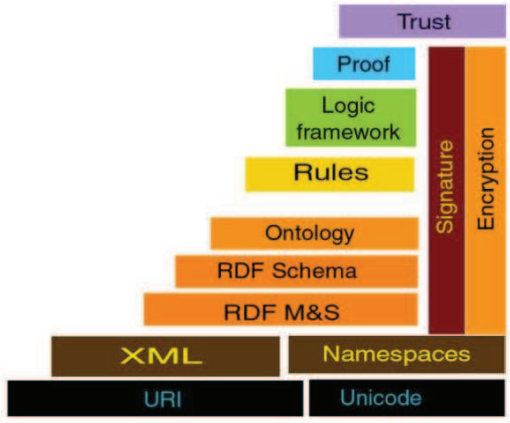


FIGURE 2.7 – Architecture du Web sémantique

XML (eXtensible Markup Language) fournit une syntaxe pour décrire la structure d'un document, pour créer et manipuler des instances de documents. Il utilise les espaces de nommage (namespaces) pour définir et identifier les balises qui forment un document. Le schéma de ce dernier permet de définir les vocabulaires pour les documents XML valides. Par contre, XML n'impose aucune contrainte sémantique concernant la spécification de ces documents. Ce premier niveau d'interopérabilité n'est pas suffisant pour que les applications puissent interpréter et manipuler les données de manière significative.

Les couches RDF M&S (RDF Model and Syntax) et RDF Schéma sont considérées comme les premières fondations de l'interopérabilité sémantique. Elles permettent de décrire les taxonomies des concepts et des propriétés. RDF [14] fournit un moyen d'insérer de la sémantique dans un document, l'information est conservée principalement sous forme de ce

12. Uniform Resource Identifier
13. Uniform Resource Locator
14. Resource Description Framework

qu'on appelle déclaration RDF. RDF Schéma (RDFS) décrit les hiérarchies des concepts, leurs relations, les propriétés et leurs restrictions de domaine.

La couche Ontologie (Ontology) décrit des sources d'information hétérogènes, distribuées et semi-structurées en définissant un consensus de domaine commun et partagé entre plusieurs personnes ou communautés. Les ontologies permettent aux machines et aux humains à communiquer avec précision en utilisant des échanges sémantiques plutôt que syntaxiques seulement. Les règles (Rules) constituent aussi un élément clé de la vision du Web sémantique. Cette couche offre les moyens d'intégration, de dérivation et de transformation de données provenant de sources multiples.

Ensuite, nous trouvons une couche logique (Logic framework) au-dessus de la couche Ontologie. Certains considèrent ces deux couches comme étant au même niveau parce que les ontologies se basent sur la logique et permettent des axiomes logiques. En appliquant la déduction logique, on peut inférer de nouvelles connaissances à partir d'une information explicitement représentée.

Les couches Preuve (Proof) et Confiance (Trust) sont les couches qui assurent la vérification des déclarations effectuées dans le Web Sémantique. Elles permettent d'avoir un environnement Web plus fiable et mieux sécurisé dans lequel des tâches d'authentification, de sécurité et de sûreté peuvent être assurée. La provenance des connaissances, des données, des ontologies ou des déductions est authentifiée et assurée par des signatures numériques. Aussi, dans le cas où la confidentialité des données et des documents est exigée, les techniques de chiffrement peuvent être utilisées.

#### 2.4.3.1 Resource Description Framework : RDF

RDF est un formalisme standard adopté par le W3C [15] pour la représentation des connaissances sur le Web. Il fournit l'interopérabilité entre les applications qui échangent des informations sur le Web et les rend compréhensibles par les machines. RDF augmente la facilité de traitement automatique des ressources Web. Il peut être utilisé pour annoter des documents écrits dans des langages non structurés ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente. La syntaxe de RDF repose sur le langage XML. XML fournit une syntaxe pour encoder des donnés tandis que RDF fournit un mécanisme décrivant leur sens. Un des buts de RDF est de rendre possible la spécification de la sémantique des données en se basant sur XML d'une manière standardisée et interopérable.

Un document RDF est un ensemble de triplets de la forme $< ressource, propriété, valeur >$ avec :

– une ressource est une entité accessible sur Internet via une URI, elle peut être un document HTML ou XML, une image, une page Web, une partie d'une page Web. . .,

15. World Wide Web Consortium : http ://www.w3.org

– une propriété définit une relation binaire entre une ressource et une valeur permettant ainsi d'associer de l'information sémantique à une ressource,
– une valeur est une ressource ou une valeur littérale (chaîne de caractères).

Une déclaration RDF spécifie la valeur d'une propriété d'une ressource. On peut la décrire comme ***propriété (ressource, valeur)***. Les éléments de ces triplets peuvent être des URIs, des littéraux ou des variables. Cet ensemble de triplets peut être représenté de façon naturelle par un multi-graphe orienté étiqueté où les éléments apparaissant comme ressources ou valeurs sont les sommets et chaque triplet est représenté par un arc dont l'origine est sa ressource et la destination sa valeur comme illustré sur la Figure 2.8. Dans cet exemple, la ressource est un document qui a comme URI *https ://novateam.safe-access.com/vault/user5921/docu87*, distribué_par est une propriété qui a pour valeur *Novapost SAS*.

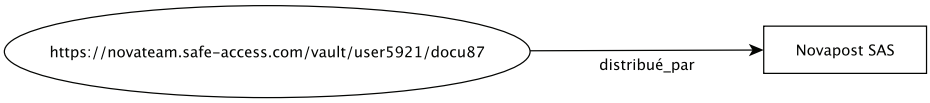


FIGURE 2.8 – Modèle de triplet en RDF

#### 2.4.3.2 RDF Schéma (RDFS)

Pour décrire n'importe quel type de connaissances à l'aide du formalisme que nous venons de décrire, il faut d'abord écrire en RDF le modèle sémantique à utiliser. Par exemple, pour décrire des connaissances en terme de concepts et de relations hiérarchisés, l'introduction des **concepts**, **relations** et des **propriétés** de subsomption et d'instanciation est nécessaire. Un schéma de base incluant les primitives sémantiques généralement utilisées a ainsi été ajouté à RDF pour constituer RDFS. Une fois ce schéma stocké sur le Web, les primitives qui y sont décrites peuvent être utilisées dans une page Web si on y inclut une référence à l'URI du schéma. Une application nécessitant l'accès à la sémantique de cette page utilisera alors le schéma d'interprétation. RDFS a pour but d'étendre RDF en décrivant de façon précise les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les instances seront des ressources.

Un schéma RDFS s'écrit toujours à l'aide de triplets RDF en utilisant deux propriétés fondamentales *subClassOf* et *type* pour représenter respectivement les relations de subsomption entre classes et les relations d'instanciation entre instances et classes. Les classes spécifiques au domaine sont déclarées comme des instances de la ressource *Class* et les propriétés spécifiques au domaine comme des instances de la ressource *Property*. Les propriétés *subClassOf* et *subPropertyOf* permettent de définir des hiérarchies de classes et de propriétés respectivement. D'autre part, RDFS ajoute à RDF la possibilité de définir les contraintes de domaine et de co-domaine de valeurs avec les attributs *rdfs :domain* et *rdfs :range*.

Pour résumer, **XML** peut être vu comme **la couche de transport syntaxique**, **RDF** comme **un langage relationnel de base** et **RDFS** comme **un langage offrant des primitives de représentation de structures ou primitives ontologiques** [Laublet *et al.*, 2002].

#### 2.4.3.3 Les ontologies

Aujourd'hui, les ordinateurs évoluent d'appareils déconnectés et isolés à des points dans un réseau universel d'échange d'informations et de transactions commerciales [Fensel *et al.*, 2000]. Ceci est dû principalement aux différentes technologies du Web qui ont cassé les barrières traditionnelles de publication, d'accès, de partage et de consommation de l'information. Si les liens hypertextes permettent aux utilisateurs de traverser cet espace d'information, que les moteurs de recherche indexent les informations et classifient leur contenu, les ontologies jouent un rôle favorisant le partage de l'information, sa structuration et sa compréhension.

L'avantage des ontologies est qu'elles permettent une compréhension partagée et commune d'un domaine donné pour un ensemble de systèmes ou d'utilisateurs. Leur développement est souvent un processus coopératif impliquant plusieurs personnes ayant des compétences différentes et complémentaires. Le terme ontologie est initialement emprunté de la philosophie signifiant explication systématique de l'existence. En informatique, une ontologie définit un vocabulaire et est similaire à un dictionnaire ou un glossaire avec une structure précise et détaillée qui permet aux machines de traiter son contenu.

Plusieurs définitions ont été donné aux ontologies ces dernières années. Nous adoptons la définition de Gruber [Gruber, 1993] : *une ontologie est une spécification explicite et formelle d'une conceptualisation partagée.* Une conceptualisation se réfère à un modèle abstrait d'un phénomène. Elle identifie les concepts relatifs à ce phénomène. Explicite veut dire que les types de concepts utilisés et les contraintes sur leur utilisation sont explicitement définis. Formelle se réfère au fait que l'ontologie doit être traitable par une machine et que la machine soit capable de l'interpréter sans aucune ambiguïté. Le terme partagé reflète l'idée que l'ontologie capitalise un ensemble de connaissances qui n'est pas restreint à un seul individu et qui est accepté par un groupe. Toutefois, la conceptualisation d'un domaine de connaissance ne peut se faire de manière non ambiguë que dans un contexte d'usage précis. Par exemple, le même terme peut désigner deux concepts différents dans deux contextes d'utilisation différents. Donc, cette modélisation n'est valable que dans un domaine de connaissance donné et pour un but donné.

Une ontologie est un outil de représentation de connaissances. Ces connaissances portent sur des objets auxquels on se réfère à travers une sémantique formelle. Les connaissances d'un domaine formulées à travers une ontologie sont transmises par les éléments suivants :

- *Concepts (ou classes)* : représentent les objets, abstraits ou concrets, élémentaires ou composites du monde réel. Les concepts sont organisés en taxonomie par l'utilisation de la relation de subsomption. Par exemple, les concepts *Acteur_Humain* et *Acteur_Non_Humain* représentent des classes différentes d'objets pour lesquelles Acteur

est un super-concept. Le concept Document_RH (Document Ressources Humaines) décrit une classe d'objets pour les documents gérés par les ressources humaines au sein d'une entreprise comme les contrats de travail ou les fiches de paie. Ce type de documents appartient à une sous-classe des documents à valeur probante *Document_VP*. Cette dernière (classe *Document_VP*) est une sous-classe du concept *Document*.

- *Relations* : représentent des liens sémantiques de la connaissance du domaine. Une relation peut être distinguée comme une propriété ou un attribut. Les propriétés décrivent des interactions entre concepts. Elles peuvent être fortement typées, c'est à dire associées à un domaine et un co-domaine précis qui permettent respectivement de spécifier les classes susceptibles d'initialiser une propriété et de contraindre les domaines de valeurs des propriétés. Dans l'exemple ci-dessous (voir Figure 2.9), la propriété *signer_électroniquement* indique une relation entre les concepts *Acteur_Non_Humain* et *Document_Valeur_Probante* et dans laquelle *Acteur_Non_Humain* est le domaine et *Document_Valeur_Probante* est le co-domaine de cette propriété. Les attributs correspondent à des caractéristiques, des spécificités particulières, attachées à un concept et qui permettent de le définir de manière unique dans le domaine. Leurs valeurs sont littérales (i.e. de type primitif) comme les chaînes de caractères ou les nombres entiers. Par exemple, le concept Document peut avoir les attributs nom, date de création, propriétaire. . ..
- *Axiomes* : constituent des assertions (ou des prédicats) acceptées comme vraies qui s'appliquent sur les classes ou les instances des classes de l'ontologie et qui permettent de restreindre les interprétations possibles d'une ontologie et/ou de déduire de nouveaux faits à partir de faits connus.
- Instances (ou objets) : représentent des instanciations des concepts. Une instance appartient à un seul concept parent et est associée à un identifiant unique permettant de la distinguer des autres instances. Sur notre exemple (voir Figure 2.9), **paie2011** est une instance du concept Document_RH.

#### 2.4.3.4 Le langage OWL (Web Ontology Language)

OWL [16] est un langage pour représenter des ontologies dans le Web sémantique. C'est une extension du langage RDFS. OWL offre aux machines de plus grandes capacités d'interprétation du contenu Web que celles permises par XML, RDF et RDFS grâce à un vocabulaire supplémentaire et une sémantique formelle. Inspiré des logiques de descriptions et successeur de DAML+OIL [Hiep, 2007], OWL fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les classes de manière plus complexe correspondant aux connecteurs de la logique de description équivalente (intersection, union, restrictions diverses. . .), les propriétés des classes définies (comme la disjonction),

16. www.w3.org/TR/owl-features

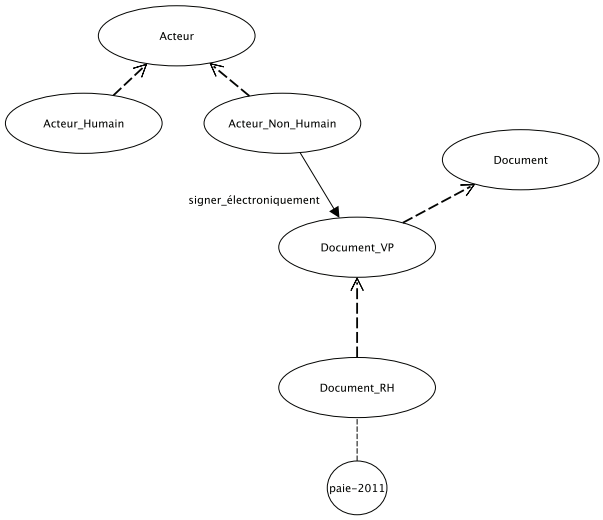


FIGURE 2.9 – Exemple d'une ontologie

la cardinalité (par exemple "exactement un"), plus des types des propriétés (propriétés d'objet ou d'annotation...), des caractéristiques des propriétés (par exemple la symétrie, la transitivité), et des classes énumérées. OWL se compose de trois sous-langages offrant une expressivité croissante [Bechhofer *et al.*, 2004] OWL-Lite, OWL-DL et OWL-Full.

- OWL-Lite : ce sous-langage ne contient qu'un sous-ensemble réduit des constructeurs. Il a la complexité formelle la plus basse et l'expressivité minimale dans la famille OWL. Il est suffisant pour représenter des thésaurus et d'autres taxonomies ou des hiérarchies de classification avec des contraintes simples.
- OWL-DL contient l'ensemble des constructeurs mais avec des contraintes particulières sur leur utilisation qui assurent la décidabilité de la comparaison de types. Par contre, la grande complexité de ce langage le rend difficilement calculable de manière efficace.
- OWL-Full : Ce sous-langage est conçu pour ceux qui ont besoin de l'expressivité maximale, de la liberté syntaxique de RDF mais sans garantie de calculabilité.

#### 2.4.3.5 Le langage SPARQL

Avec toutes ces évolutions dans les technologies de représentation de connaissances, le besoin de technologies d'interrogation adaptées est devenu crucial. Le langage de requêtes SQL, standard d'interrogation des bases relationnelles n'est pas adapté pour l'interrogation sémantique. D'autres langages de requêtes comme XPATH, RQL et TRIPLE existent. Cependant, soit ils ne sont pas adaptés pour effectuer des requêtes sémantiques de façon native, soit ils n'ont pas évolué pour atteindre le stade de maturation et de standardisation. Pour pallier à ce manque, le groupe de travail RDF Data Access du W3C a proposé SPARQL [Prud'hommeaux et Seaborne, 2008], un langage de requêtes et un protocole pour l'accès aux données RDF. Ce langage se base sur la correspondance des patrons de

graphe (graph patterns matching). Le patron de graphe le plus simple est le patron de triplets (comme un triplet en RDF) mais il possède la capacité d'exprimer des variables de requête dans les positions du sujet, de la propriété ou de l'objet d'un triplet. D'autre part, SPARQL intègre des balises spécifiques telles que le patron de graphe optionnel (OPTIONAL), l'union (UNION) et l'intersection des patrons, le filtrage (FILTER) ou les opérateurs de comparaison des valeurs. Ils permettant d'effectuer des requêtes plus efficaces et flexibles. Nous présentons un exemple de requête en SPARQL (voir Exemple de code 2.1, Exemple de code 2.2 et Exemple de code 2.3) qui demande toutes les ressources (i.e. ?document) de type (i.e. rdf :type) "Document" selon la définition de l'ontologie nova (i.e. nova :Document).

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX nova: <http://www-inf.int-evry.fr/~sakka_mo/domainModels/nova.owl#>
3 SELECT ?document
4 WHERE {
5       ?document rdf:type nova:Document .
6 }
```

Exemple de code 2.1: Requête SPARQL de sélection

```
1 PREFIX nova: <http://track.novapost.fr/DM/probative.owl>
2 PREFIX mdm: <http://\\
3 .novapost.fr/DM/mdm.owl>
4 SELECT  ?documentName
5 FROM <http://track.fr/idms/novapost.rdf>
6 WHERE   { ?document mdm:hasName ?documentName .
7           ?document nova:distributionDate ?date .
8               FILTER (
9               ?date > ''2011-10-1''^^xsd:date &&
10              ?date < ''1012-10-30''^^xsd:date )
11                }
```

Exemple de code 2.2: Requête SPARQL de type "Filter"

```
1 PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
2 PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
3 CONSTRUCT   { <http://example.org/person#Alice> vcard:FN ?name }
4 WHERE       { ?x foaf:name ?name }
```

Exemple de code 2.3: Requête SPARQL de type "Construct"

#### 2.4.3.6 Systèmes de gestion de provenance basés sur les technologies du Web sémantique

Après cette introduction sur le Web sémantique et ses technologies, nous reprenons dans cette partie les systèmes de gestion de provenance basés sur ces technologies. Dans [Sahoo *et al.*, 2008], les auteurs ont proposé une approche de gestion de provenance, séparant ce qui est relatif à la provenance de ce qui est spécifique au métier en se basant sur les ontologies. Le framework PrOM [Sahoo *et al.*, 2009] développant cette approche propose un modèle de provenance basé sur une ontologie minimale appelée *Provenir*. Cette ontologie peut être enrichie et étendue pour répondre parfaitement aux exigences spécifiques d'un

métier donné. Ceci permet de réduire le travail de modélisation et assure un niveau minimal d'interopérabilité entre les différents modèles de provenance. Par exemple, l'ontologie *Trident* étend l'ontologie *Provenir* pour modéliser la provenance dans le domaine de l'océanographie (voir Figure 2.10).

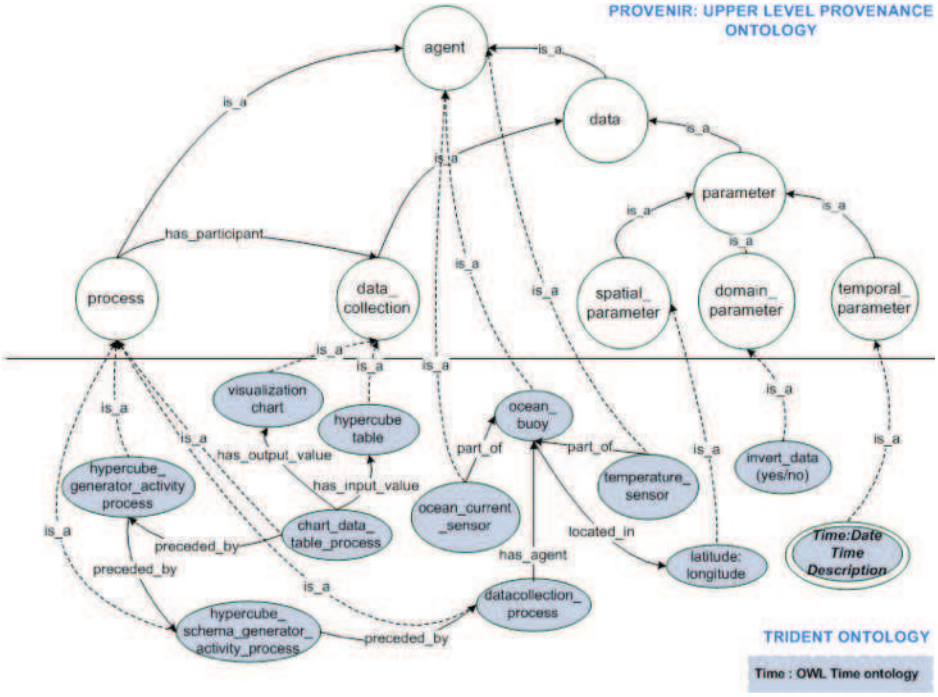


FIGURE 2.10 – L'ontologie Trident héritant de l'ontologie Provenir [Sahoo *et al.*, 2009]

Ce travail détaille aussi les caractéristiques et les propriétés des requêtes de provenance et identifie trois catégories de requêtes. Des opérateurs de requêtes spécifiques pour ces catégories ont été développé et un mécanisme de traitement adéquat pour chaque catégorie de requête a été proposé. Ces opérateurs forment aussi une base pour la construction d'une infrastructure scalable pour le stockage et l'interrogation de la provenance. Nous jugeons que ce travail propose une approche de modélisation extensible et adéquate pour la provenance. Ces opérateurs ont été implémenté et testé sur un RDF store Oracle. Pour résoudre les limitations rencontrées pour les requêtes de provenance complexes, des optimisations ont été introduites par la création de vues de provenance matérialisées (MPV : Materialized Provenance Views). Ce framework a été développé et testé entièrement dans le cadre de la gestion de provenance pour les applications scientifiques.

Dans un autre travail, Freitas et al. s'intéressent aux applications qui génèrent et échangent des données liées et considèrent la provenance comme un élément pilier pour la qualité des données sur le Web [Freitas *et al.*, 2010]. Ce travail propose un framework appelé Prov4J [17] pour la gestion de la provenance dans le Web. Il se base sur les technologies

17. www.prov4j.org

du Web sémantique pour l'implémentation d'un PMS et discute les composants logiciels clés pour la collecte et la consommation de la provenance.

Pour la modélisation de la provenance, Prov4j définit une ontologie appelée W3P [18] qui est compatible avec le modèle OPM et utilise des vocabulaires standards comme Dublin Core [19] et ChangeSet [20]. Prov4J propose d'utiliser les URIs pour référencer les données de façon homogène ce qui assure l'unification du système de référencement. Aussi, ce framework possède l'avantage de pouvoir consommer et utiliser la provenance à partir de plusieurs types de sources. Une source pour Prov4J peut être un service SPARQL (SPARQL endpoint), des données RDF publiées en utilisant les principes de publication des données liées [21] ou un descripteur de provenance qui est un ensemble de données RDF embarquées dans une ressource donnée.

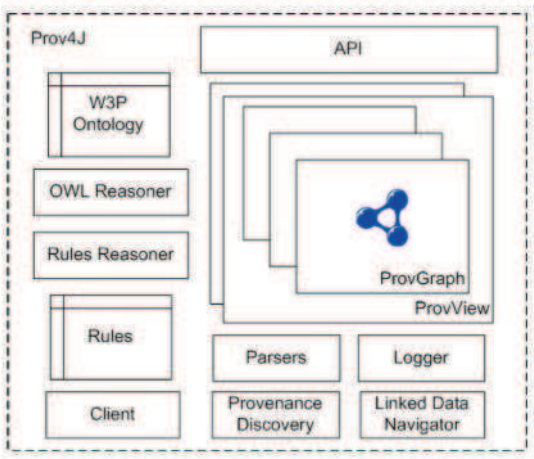


FIGURE 2.11 – Les composants de Prov4J [Freitas *et al.*, 2010]

Prov4J traite aussi le couplage de la provenance avec les sources d'information et sa découverte automatique. Il propose un mécanisme de découverte automatique pour un ensemble de sources d'information comme les pages Web (ou des éléments dans des pages Web), les services SPARQL, les fichiers RDF ou bien les URIs. Les mécanismes de découverte de provenance supportés dans Prov4J sont :

- les sitemaps sémantiques et les robots : ils sont utilisés pour découvrir les descripteurs de provenance d'un ensemble de données à partir d'un nom de domaine,
- les données de provenance liées (Linked provenance data) : les descripteurs de provenance peuvent être publiées comme des données liées de deux manières. La première est de considérer qu'une URI représente un artifact et est reliée directement aux données de provenance, la deuxième est d'associer la ressource avec sa provenance

18. www.prov4j.org/w3p/schema
19. www.dublincore.org
20. www.vocab.org/changeset
21. http ://www.w3.org/DesignIssues/LinkedData.html

en utilisant une propriété (comme la propriété *provenance* dans l'ontologie W3P) qui relie une URI à un descripteur de provenance,
- RDFa embarqué [22] : la provenance est embarquée dans les pages HTML en RDFa.
- POWDER [23] : utilisation de ce protocole qui est une recommandation W3C [24] pour décrire les ressources Web.

Son architecture (voir Figure 2.11) est modulaire et possède beaucoup de similarités avec l'architecture proposée par Groth [Groth *et al.*, 2006]. Elle se base sur un module pour les règles et le raisonnement sémantique, un module pour la découverte de provenance, un module de stockage, un module de navigation, des parsers et un logger (pour importer la provenance dans le module de stockage). Une API pour la consommation de la provenance a été aussi développée et contient une interface de requêtes pour pouvoir effectuer les requêtes dans un langage autre que SPARQL.

Dans [Hartig et Zhao, 2010] un framework pour la gestion de provenance des données liées sur le Web est proposé. Il traite les questions de publication et de consommation de provenance sur le Web. La contribution de se travail par rapport aux autres frameworks est qu'il traite d'une manière détaillée deux dimensions de gestion de provenance qui n'étaient pas très bien développées et qui sont la création et l'accès aux données liées. Pour ce faire, les auteurs ont proposé un vocabulaire de provenance permettant de capturer les informations sur la création et l'accès aux données liées. Nous pouvons remarquer une grande similarité de ce vocabulaire avec OPM : l'équivalent des trois entités OPM de base (Agent, Process, Artifact) existe dans ce vocabulaire. Elles sont remplacées respectivement par (Actor, Execution, Artifact). Comme OPM, ce framework est aussi extensible et offre la possibilité de personnaliser le vocabulaire avec de nouveaux attributs en plus des attributs de base (appelés attributs universels). Ce framework se base sur la consommation de méta-données qui existent sur le Web pour extraire les informations sur la provenance des ressources. En effet, les données sont publiées sur le Web et les accès se basent généralement sur des URIs via le protocole HTTP. Il devient ainsi possible d'interroger un service WHOIS [Hartig, 2009] pour obtenir des informations de provenance, d'utiliser les sitemaps sémantiques ou bien le protocole POWDER [25] qui permet de décrire des données sur le Web en utilisant des mots clés (tags) ou RDF. Ce travail a proposé aussi des mappings entre le vocabulaire proposé et d'autre vocabulaires qui étaient utilisés pour la description de provenance comme Dublin core, FOAF (Friend Of A Friend), SIOC (Semantically-Interlinking Online Communities), OMV (Ontology Metadata Vocabulary), Proof Markup Language[Silva *et al.*, 2006], Changeset Vocabulary et l'ontologie de provenance Ouzo [Zhao, 2007]. Ces mappings favorisent l'interopérabilité sémantique entre les différents vocabulaires.

22. www.w3.org/TR/xhtml-rdfa-primer
23. Protocol for Web Description ressources : www.w3.org/TR/powder-dr
24. www.w3.org
25. Protocol for Web Description Resources

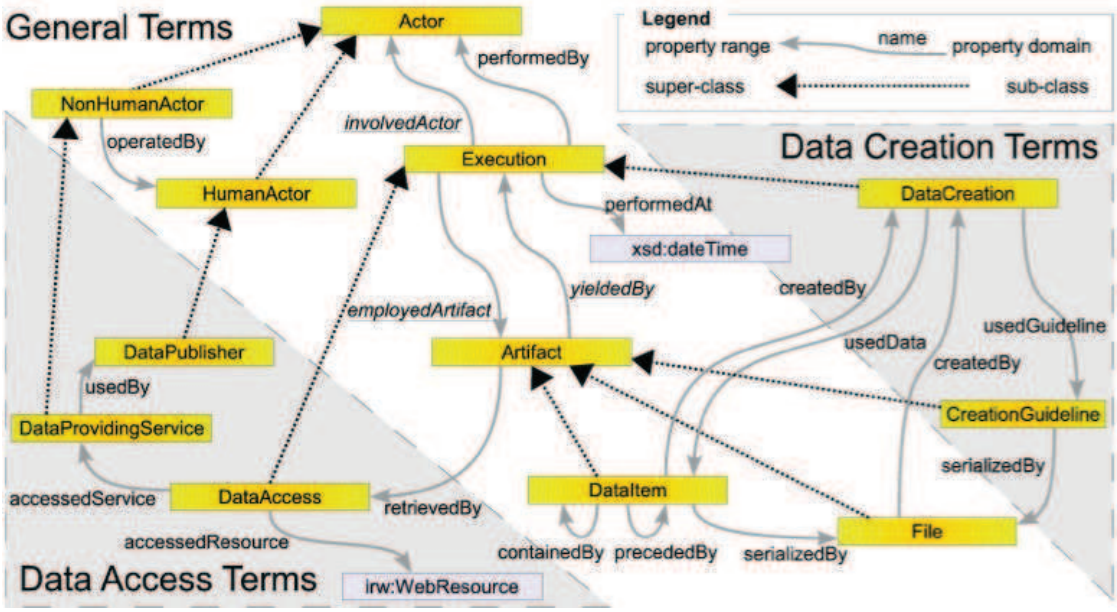


FIGURE 2.12 – Classes et propriétés du vocabulaire de provenance proposé par Hartig pour la dimension *Data Access* [Hartig et Zhao, 2010]

### 2.4.4 Provenance dans le domaine de la gestion électronique de documents

Pour le domaine de la GED (Gestion Électronique de Documents) et des services d'archivage légal qui constituent notre cadre de travail principal et afin de mieux appréhender l'existant et les besoins en terme de provenance, nous avons effectué des recherches sur les outils et les techniques existantes.

Les recommandations en la matière comme la norme d'archivage légal NF Z42-013 [AFNOR, 2009] de l'AFNOR [26] ou le standard européen MoReq2 [MoReq2, 2008] exigent d'assurer la pérennité des données sur le long terme ainsi que leur valeur probante. Si l'intégrité technique des données échangées et archivées peut être assurée par les empreintes numériques et si la non répudiation peut être assurée par des procédés de signature électronique, leur traçabilité reste perfectible. Aujourd'hui, les moyens techniques de génération, de conservation et de vérification des empreintes et des signatures électroniques existent. Les moyens techniques de conservation à long terme des données et des documents existent également. Par contre, on ne dispose pas des dispositifs permettant de démontrer et de garantir dans le temps, la validité de la signature au moment où celle-ci a été générée, ni l'authenticité d'une entité au moment où elle a effectué une action. Pour assurer la valeur probante, ces plateformes et services doivent générer et garder des traces permettant d'assurer la validité de la signature électronique, de vérifier l'origine et l'identité des acteurs effectuant des actions sur les documents et d'identifier les procédures et les moyens techniques intervenants dans le cycle de la vie du document.

26. Agence Française de NORmalisation : www.afnor.org

Par exemple, la GED Documalis [27] incorpore des fonctionnalités permettant de tracer l'historique des opérations effectuées sur un document. Cette traçabilité est garantie depuis la capture initiale du document sur la plateforme. Elle s'appuie sur un format normalisé appelé *ZDOC* qui est un format d'échange universel utilisé par Documalis pour le transfert des documents. Ce format intègre dans un container crypté le document, ses index au format XML ainsi que toutes les informations de traçabilité. Toutes les opérations de déplacement, de transformation et de modification de version du document au sein de la plateforme sont capturées.

Les outils de gestion et d'automatisation des processus documentaires intègrent aussi la dimension provenance. Par exemple, Track & Trace est une solution de traçabilité documentaire développé par Nirva [28]. Cet outil vise à simplifier les processus pour les spécialistes du document ainsi que pour les utilisateurs finaux en prenant en compte les aspects de traçabilité. Cette solution de traçabilité en temps réel peut être utilisée par les utilisateurs finaux (comme les gestionnaires de comptes), les responsables techniques (comme les gestionnaires de production) ou les clients finaux. Elle est composée de quatre composants principaux :

- un module d'administration basé sur une interface Web qui permet aux entreprises de définir ce qui doit être tracé, avec quel niveau de granularité et le type de rapport souhaité. Track & Trace génère alors automatiquement un modèle de données et met en forme le portail Web pour le reporting standard (menus, sous-menus, écrans ...).
- un module de chargement : collecte les données tout au long du processus de production des documents et alimente le référentiel de données, quel que soient les éléments qui constituent l'infrastructure de production de documents (outil de composition, imprimantes ...).
- un modèle de données disponible pour plusieurs types de base de données (Oracle, DB2, SQL Server ...), complètement ouvert et qui peut être utilisé à d'autres fins. Si un modèle de données est déjà disponible, la génération automatique d'un nouveau modèle n'a pas lieu et les rapports sont constitués à partir du modèle de données déjà existant.
- un outil de traçabilité pour restituer l'information sous la forme d'interfaces. Ce module peut intégrer un générateur de rapports et alimenter une solution d'analyse décisionnelle.

Pour la gestion documentaire à valeur probante, la plupart des acteurs du marché comme Opentrust [29], Dictao [30], Linagora [31], Cryptolog [32], CDC Arkhineo [33], Certeurope [34] se basent

27. www.documalis.com
28. www.nirva-systems.com
29. www.opentrust.fr
30. www.dictao.fr
31. www.linagora.com
32. http ://Web.cryptolog.com
33. www.cdcarkhineo.com
34. www.certeurope.fr

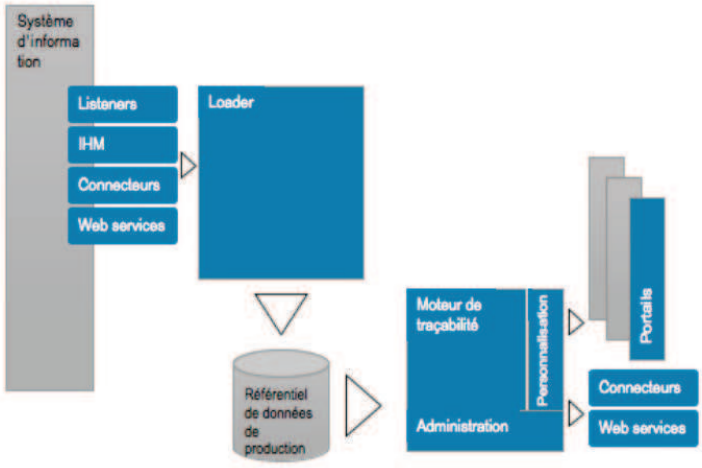


FIGURE 2.13 – Les modules de Track & Trace

sur des « enveloppes de preuve » contenant des signatures et des jetons d'horodatage. Ces méta-données peuvent authentifier l'auteur d'une action sur un document, certifier la date et l'heure de sa création ou de sa modification, vérifier son intégrité et qu'il n'est pas altéré. Ces systèmes utilisent des serveurs de signature électronique, des serveurs d'horodatage ainsi que des systèmes de stockage et d'archivage répondant aux normes et aux standards en vigueur. Cependant, le lien entre ces éléments de preuve dispersés, leur pérennité dans le temps et leur gestion ne s'effectue pas dans le cadre d'une politique globale qui prend en compte les spécificités de chaque entité intervenante. Ainsi, le fil conducteur entre ces différentes éléments de preuve risque d'être perdu.

### 2.4.5 Autres contraintes sur les systèmes de gestion de provenance

#### 2.4.5.1 Sécurisation de la provenance

La sécurisation de la provenance, malgré son importance n'a pas été suffisamment traitée. Étant donné que la provenance est un élément de valeur probante, sa sécurisation est cruciale. Une provenance erronée ou falsifiée n'a aucune valeur ajoutée et peut conduire à la destruction de la valeur de l'information et de sa fiabilité.

Le travail le plus abouti sur la sécurisation de la provenance que nous avons trouvé est celui de Hasan et al. [Hasan *et al.*, 2009]. Ce travail propose un schéma pour la sécurisation de la provenance (voir Figure 2.14). Il se base sur le chainage des enregistrement de provenance et le calcul d'une somme de contrôle. Le cryptage est aussi utilisé pour assurer la confidentialité de la provenance qui ne sera accessible que pour les profils autorisés. Ce travail a été validé par des implémentation du schéma de sécurisation proposé pour tracer la provenance des fichiers (en écriture, lecture et accès) dans un système de stockage et au niveau d'un système d'exploitation. Il propose les fondations nécessaires pour la sécurisation de la provenance mais il est centré uniquement sur les systèmes monolithiques

et son mode d'intégration est complexe et intrusif. Son développement sur les systèmes ouverts, hétérogènes et distribués ainsi que la réduction de son coût d'intégration nécessitent d'être mieux développés.

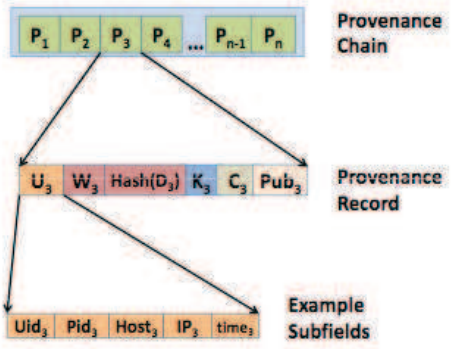


FIGURE 2.14 – Schéma de sécurisation de la provenance proposé par Hasan et al. [Hasan *et al.*, 2009]

#### 2.4.5.2 Interrogation de sources de provenance distribuées

Un autre aspect qui n'a pas été étudié et approfondi dans le contexte de gestion de provenance est celui de l'interrogation de sources de provenance distribuées. Certes, l'interrogation de base de données distribuées est un sujet qui a été largement traité [Ozsu et Valduriez, 2011]. Cependant, les requêtes de provenance possèdent leurs propres caractéristiques et spécificités. En effet, l'évolution des architectures orientées services, du Cloud computing et des technologies du Web rendent les sources de provenance multiples et hétérogènes. De nouvelles techniques pour l'interrogation de sources de provenance hétérogènes et distribués doivent être développées pour répondre aux besoins spécifiques de ce contexte.

Dans [Zhao *et al.*, 2011b], les auteurs ce sont intéressés à cette problématique d'interrogation de sources de provenance distribuées. Le système d'interrogation proposé dans ce travail se base sur un service d'index qui contient des méta-données sur les artifacts et les sources de provenance. Cependant, il se limite à quelques types de requêtes (les requêtes de chemin, forward et backward n'ont pas été traité). Aussi, il n'adresse que des requêtes spécifiques, fortement liées aux workflows de prévision de production et n'est pas générique. Les questions de scalabilité de l'architecture proposée en stockage et en interrogation n'ont pas été traité.

### 2.4.6 Synthèse

Nous nous sommes intéressés dans cette deuxième partie de notre étude bibliographique aux systèmes de gestion de provenance. Les travaux que nous avons présenté couvrent

plusieurs domaines comme les bases de données, les systèmes de workflows, le Web et les données liées ainsi que les systèmes de gestion documentaire.

Les systèmes de gestion de provenance pour les bases de données comme [Widom, 2005] sont conçus pour des données stockées dans des bases de données relationnelles ou XML et qui résultent de l'exécution de requêtes (ou de procédures stockées). Ces systèmes se basent sur l'inversion de requêtes et ne sont pas par conséquent utilisables dans notre contexte.

Pour les systèmes de gestion de provenance [Sahoo *et al.*, 2009, Stevens *et al.*, 2003], une grande partie se focalise sur des besoins de provenance spécifiques à un domaine d'application (biologie, science de la terre ...), un type de système (base de données, système de workflow ...). Ces systèmes, issus principalement de la communauté de workflow scientifique sont matures, répondent aux besoins de cette communauté avec des modèles de provenance ayant un grand niveau de détail. Dans un cadre industriel, ProM et myGrid ont été utilisés avec de hautes exigences en terme de charge et de scalabilité. Par contre, ils ne sont pas conçus pour être génériques ou extensibles. Ils ne supportent pas l'utilisation d'autres modèles de représentation de provenance et sont basés sur des modes de collecte assistés nécessitant un effort d'intégration. Les systèmes de gestion de provenance les plus aboutis que nous avons trouvés et qui couvrent le cycle de vie de la provenance (collecte, sauvegarde, gestion et interrogation) sont PASOA [Groth *et al.*, 2004] et l'architecture proposée par Groth et al. [Groth *et al.*, 2006]. Ces deux systèmes peuvent être utilisés dans des architectures à base de services.

Freitas [Freitas *et al.*, 2010] et Hartig [Hartig, 2009] ont exploré la provenance dans le Web de données liées et ont proposé deux frameworks pour la gestion de provenance. Cet espace de création, de publication et de consommation de données ouvert ne cesse d'évoluer ce qui rend la provenance une information très importante, voir cruciale. Ces deux frameworks maximisent l'utilisation des outils du Web sémantique. Ils utilisent les ontologies pour la modélisation et l'inférence, RDF pour avoir un format de représentation unique, les URIs pour l'identification des ressources et les techniques d'interrogation sémantique comme SPARQL. *RDF_*PROV est aussi un système de gestion de provenance basé sur les outils du Web sémantique pour la modélisation et la représentation. Par contre, il propose des mapping vers des systèmes de base de données relationnelles pour le stockage afin de pouvoir effectuer des interrogations avec SQL au lieu de SPARQL.

Le stockage sécurisé de la provenance n'a pas été traité d'une manière approfondie. Dans [Hasan *et al.*, 2009], les auteurs se sont intéressés à la provenance dans les bases de données et les systèmes de stockage et à sa sécurisation. Il a proposé un modèle pour la sécurisation des éléments de provenance en les chainant et en utilisant les techniques de hachage et de calcul de sommes de contrôle.

Pour l'exploitation de la provenance, nous avons identifié principalement trois types : l'interrogation via des API, les requêtes et les graphes visuels. Cependant, les questions d'interrogation distribuée de la provenance n'ont pas été suffisamment traitées et méritent encore plus d'intérêt. Dans [Zhao *et al.*, 2011b], les auteurs ce sont intéressé à l'interrogation

de la provenance dans un environnement distribué. Cependant, ce travail se limite à quelques catégories de requêtes de provenance (il ne traite pas les requêtes de chemin, forward et backward) et n'adresse pas les questions de scalabilité du système proposé dans les environnements de production.

A travers cette étude, nous constatons que la plupart de ces PMSs sont fortement liés à leurs domaines d'utilisation. Nous constatons aussi que les questions de scalabilité de PMSs, de sécurisation de la provenance et d'interrogation de sources de provenance distribuées n'étaient pas suffisamment traitées. Ainsi, nous identifions à travers cette étude un ensemble d'exigences qu'un système de gestion de provenance doit assurer et qui sont principalement :

1. support de fonctions d'administration pour la collecte de provenance (ajout de nouvelles source, suppression de sources existantes ...),
2. support de fonctions d'administration permettant la gestion des modèles de provenance (ajout de nouveaux modèles, définition de relations de spécialisation/généralisation entre modèles...),
3. stockage et préservation de la provenance à long terme,
4. système d'interrogation expressif,
5. traitement de requêtes dans un contexte distribué : plusieurs sources de provenance qui peuvent être hétérogènes interviennent,
6. passage à l'échelle en stockage et en interrogation.

Ces exigences sont résumés dans un tableau récapitulatif (voir Table 2.2) qui décrit chacune d'entre elle et spécifie les travaux de l'état de l'art qui la prennent en compte.

A travers notre étude bibliographique, nous pouvons confirmer qu'il n'existe aucune approche de modélisation et de conception de PMS qui satisfait les exigences que nous avons exposé dans notre introduction générale. Certes, des modèles couvrant un certain nombre de ces exigences comme OPM existent mais ne couvrent pas nos exigences d'un point de vue conception de PMS (comme le stockage à long terme, l'interrogation distribuée et la scalabilité). Inversement, des architectures de PMS configurables, extensibles ou à capacité industrielle existent mais ne couvrent pas nos exigences en terme de modélisation. Les exigences non couvertes par ces systèmes sont principalement les aspects de corrélation, de fusion et de création des vues métier.

Nous allons nous intéresser dans le cadre de notre travail de thèse à toutes ces exigences (celles de l'aspect de modélisation ainsi que celles de l'aspect de conception de PMSs) sauf l'aspect de sécurisation de la provenance. Pour répondre à toutes nos exigences, nous proposons de nous baser sur les points forts des approches de modélisation ainsi que ceux des architectures de PMS existantes et de les coupler. Pour répondre à toutes les exigences de modélisation, nous allons proposer une approche globale pour la modélisation de la provenance basée sur un modèle générique et extensible. Cette approche vise à assurer l'interopérabilité syntaxique et sémantique entre les modèles de provenance, permettre la

| Type d'exigence | Description | Exemple de PMS |
|---|---|---|
| Fonctions de gestion de collecte | permettent de collecter la provenance depuis plusieurs sources, d'ajouter de nouvelles sources et de supprimer des sources qui existent. | myGrid, PASOA, Groth et al. [Groth *et al.*, 2006], Prov4J. |
| Fonctions de gestion de modèles | permettent de créer de nouveaux modèles de domaine, de les relier à des modèles existants (spécialisation, généralisation), de supprimer des modèles existants. | PrOM, Hartig et Zhao [Hartig et Zhao, 2010]. |
| Fonctions de stockage à long terme | assurent le stockage des données de provenance sur le long terme ainsi que leur pérennité. | RDFProv, Prov4J, Track & Trace. |
| Système d'interrogation expressif | permet d'exprimer des requêtes de provenance avec des détails et des contraintes métier spécifiques. | RDFProv. |
| Support pour le traitement de requêtes dans un environnement distribué | permet de traiter des requêtes de provenance sur plusieurs sources distribuées. | Zhao et al. [Zhao *et al.*, 2011b]. |
| Passage à l'échelle | assure le stockage de grandes volumétries de provenance et permet de les interroger de manière efficace | PrOM. |
| Sécurisation de la provenance | assure que la provenance est intègre lors de sa collecte et le long de son cycle de vie | [Hasan *et al.*, 2009]. |

TABLE 2.2: Exigences sur les systèmes de gestion de provenance

corrélation et la fusion de ces modèles une fois instanciés et supporter la création de vues métier sur les données de provenance en se basant sur les technologies du Web sémantique. Ensuite, nous proposons de traiter les questions relatives à la conception de PMS en proposant deux types d'architectures mettant en œuvre notre approche de modélisation. La premier type d'architecture est proposé selon une vision centralisée. La deuxième type est distribué et assure une vision plus globale et décloisonnée entre plusieurs sources de provenance distribuées.

## 2.5 Conclusion

Dans ce chapitre, nous avons d'abord présenté des notions terminologiques reliées à la provenance : ses définitions, ses cas d'utilisation et ses modes de collecte. Ensuite, nous nous sommes intéressés à la modélisation de la provenance et aux différents travaux proposant des modèles de provenance. L'étude des différents modèles proposés nous confirme que plusieurs modèles sont dépendants des modèles métiers des systèmes sources comme dans [Buneman *et al.*, 2000] où le modèle proposé n'est utilisable que pour la gestion de provenance pour les bases de données. Cette étude nous a permis de confirmer que le modèle OPM est un modèle générique, extensible et adresse des problématiques d'hétérogénéité syntaxiques entres des sources de provenance hétérogènes. Cependant, la gestion de profils, leurs règles d'extensions, leur sémantique et leur interopérabilité est la seule limite que nous soulignons pour OPM.

Ensuite, nous nous sommes intéressés aux systèmes de gestion de provenance dans le domaine des bases de données, de l'E-Science et des workflows scientifiques, du Web et des données liées et des systèmes de gestion électronique de documents. A partir de cette étude, nous remarquons que la majorité des PMSs n'est pas générique. Dans le domaine du Web et des données liées, les sources de provenance sont souvent très hétérogènes ce qui introduit de nouveaux défis. L'utilisation des technologies du Web sémantique comme RDF pour l'intégration des données, les ontologies pour la modélisation des concepts et des propriétés métier et SPARQL pour l'interrogation nous semble une approche prometteuse mais limitée pour assurer la scalabilité souhaitée.

Pour les systèmes de GED, l'exigence de sauvegarder et de pérenniser les traces des opérations sur les documents est préconisée par les standards de gestion et d'archivage de documents[MoReq2, 2008], [AFNOR, 2009]. Cependant, les solutions proposées aujourd'hui par les éditeurs restent limitées à la génération des éléments de preuve (signature électronique, jetons d'horodatage...). Le point faible de ces solutions est qu'elle ne proposent pas une gestion de ces éléments de preuve dans une approche globale qui prend en compte les nouvelles architectures des plateformes de GED qui font intervenir plusieurs tiers sur des cycles de vie de plus en plus complexes.

L'interrogation de la provenance selon plusieurs visions et contraintes métier n'a pas été considérée. Il est de même pour l'aspect distribué de l'interrogation de sources multiples et hétérogènes ce qui nous incite à proposer de nouvelles approches pour la modélisation des aspects métier dans les requêtes ainsi que des techniques d'interrogation plus efficaces et adaptées au contexte multi-source. Nous considérons aussi que la question de scalabilité du PMS (en stockage et en exploitation) est un point central vu les volumétries gérées et les exigences attendues en terme de performance sur ces systèmes.

Dans le chapitre suivant, nous allons traiter les questions relatives à la modélisation de la provenance que nous avons introduit dans notre introduction générale. Il s'agit de proposer une approche de modélisation permettent de traiter les questions l'interopérabilité

syntaxique et sémantique entre des modèles de provenance hétérogènes issus de plusieurs systèmes. Il s'agit aussi de traiter l'intégration des concepts et de propriétés métier dans les modèles de provenance tout en assurant une approche de modélisation générique. Cette approche traite aussi la corrélation et la fusion entre modèles de provenance instanciés ainsi que la création et la gestion des vues métier. Dans notre approche, nous ne traitons pas l'aspect de sécurisation de la provenance et nous considérons que cet aspect peut être intégré sur le modèle que nous allons proposer. Pour ce faire, nous allons exploiter les points forts des technologies du Web sémantique pour proposer une approche de modélisation de provenance, globale et cohérente permettant de répondre à nos exigences.

*Chapitre 3*

# Un cadre sémantique pour la modélisation de la provenance

*"Ce n'est point dans l'objet que réside le sens des choses, mais dans la démarche."*

Antoine de Saint-Exupery, "Citadelle"

## Sommaire

## 3.1 Introduction

Ce chapitre présente notre approche pour la modélisation de la provenance. Partant des motivations et des exigences annoncées, nous présentons un cadre (de l'anglais Framework) qui propose une approche globale et cohérente pour la modélisation de la provenance basée sur les technologies du Web sémantique.

Après notre étude bibliographique, nous avons identifié l'ensemble des propriétés qu'un modèle de provenance doit avoir et qui sont principalement :

- prendre en compte l'hétérogénéité syntaxique et sémantique des modèles de provenance,
- pouvoir exprimer des concepts et des propriétés métier spécifiques à un domaine,
- pouvoir corréler des données de provenance distribuées pour retrouver l'historique complet d'un objet numérique et générer des nouvelles informations qui ne sont pas explicites si ces données et les sources qui les hébergent sont considérées séparément,
- pouvoir exprimer et formuler des requêtes de provenance en intégrant des contraintes spécifiques à un domaine.

Le reste de ce chapitre est organisé comme suit. Dans la section 3.2 nous présentons un aperçu général de l'approche de modélisation que nous proposons. Cette approche se base sur un modèle minimal et sur un ensemble de modèles métier appelés modèles de domaine. Ces deux modèles sont détaillés respectivement dans les sections 3.3 et 3.4. Dans la section 3.5, nous présentons comment nous traitons l'aspect multi-sources et comment nous adressons les problématiques de compatibilité et de cohérence entre les modèles et les instances dans un contexte distribué. Enfin, nous concluons dans la section 3.6.

## 3.2 Un cadre sémantique pour la gestion de provenance

Généralement, la provenance d'un document est stockée sur une ou plusieurs sources. Une application peut générer la provenance dans une représentation générique qui n'est pas riche sémantiquement (log d'un serveur Web par exemple) ou sous une forme personnalisée et plus riche via des services dédiés ou bien en définissant une structure et une stratégie de création de provenance qui sera utilisée lors du développement des applications et des services. Pour pouvoir répondre à des questions sur le cycle de vie et la provenance des objets manipulés par ces applications, les différentes sources de provenance doivent être connues, associées à leurs applications sources et corrélées.

Pour pouvoir répondre aux exigences relatives à la modélisation de la provenance que nous avons identifié dans notre introduction générale, nous proposons un cadre pour la modélisation de la provenance offrant deux degrés de libertés : 1) assurer une interprétation commune de la provenance provenant de différentes sources hétérogènes en se basant sur un modèle commun. Ainsi $n$ sources hétérogènes peuvent être interprétées avec ce modèle commun et 2) assurer l'interprétation des mêmes données de provenance avec des modèles

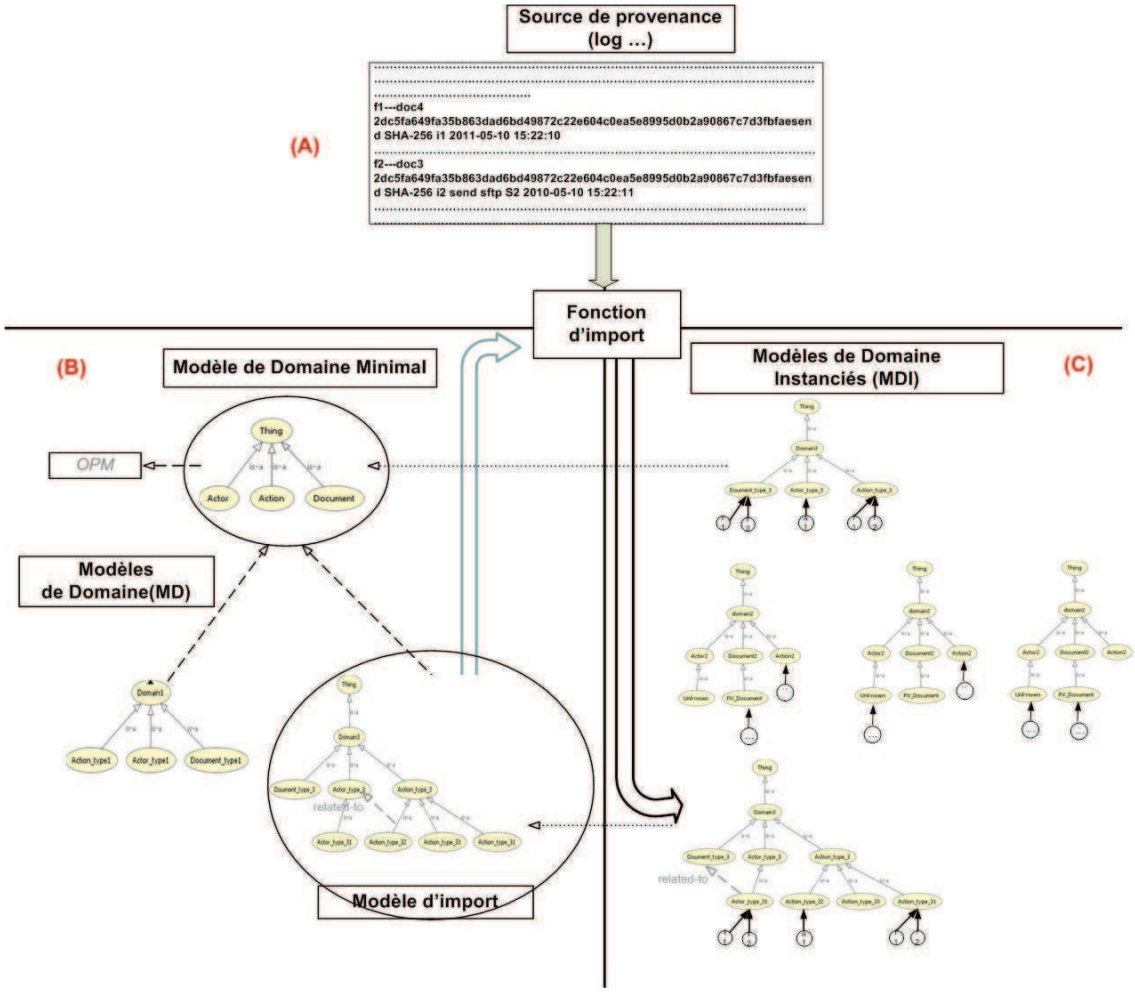


FIGURE 3.1 – Un aperçu global de notre approche

métier différents. A ce niveau, nous voulons assurer plusieurs niveaux d'interprétations pour les mêmes données de provenance.

En effet, nous voulons pouvoir fournir à des utilisateurs ayant des visions différentes sur les applications et donc des modèles métier différents la possibilité d'interpréter chacun avec son propre modèle les données de provenance d'une même application créant ainsi des vues métier sur la provenance. Par exemple, la provenance divulguée au propriétaire d'un document archivé dans un coffre fort électronique n'est pas identique à celle qui sera accessible à un auditeur qui réalise un audit sur ce même service d'archivage. Dans ce cas, nous avons deux vues métier sur les données de provenance de ce service d'archivage, une vision utilisateur (qui correspond à un modèle métier utilisateur) et une vision auditeur (qui correspond à un modèle métier auditeur).

Aujourd'hui, la multiplication des sources de provenance et l'hétérogénéité de leurs données (au niveau syntaxique et sémantique) rendent les problématiques de modélisation, d'intégration et d'interrogation de provenance des problématiques complexes et d'actualité.

Les technologies du Web sémantique facilitent la possibilité d'effectuer l'intégration des données venant de sources hétérogènes et de les lier. Avec ces technologies, la modélisation des données peut être effectuée avec des ontologies de domaine qui permettent de définir et de partager des concepts et des propriétés spécifiques à un domaine. Ces technologies proposent aussi des langages de requêtes sémantiques comme SPARQL qui permettent de formuler des requêtes sémantiques basées sur les ontologies.

Notre approche se base sur un modèle minimal que nous appelons MDM (Modèle de Domaine Minimal) permettant de ramener tout type de données de provenance issues de n'importe quelle source à un niveau d'interprétation minimal. Ceci vise à résoudre les problèmes d'interopérabilité syntaxique entre les sources. Ensuite, l'idée consiste à pouvoir spécialiser ce modèle minimal selon le domaine métier. Ces spécialisations sont définies comme des ontologies de domaine et sont appelées modèles de domaine (MD) (voir Figure 3.1 - partie B). Pour pouvoir les interroger, ces MDs doivent être instanciés. Cette instanciation est effectuée via une fonction d'import écrite par l'utilisateur souhaitant gérer la provenance de ses données et lier sa (ou ses) sources de provenance à d'autres sources. Cette fonction d'import prend en entrée un modèle de domaine appelé $MD_{import}$ qui est le modèle de domaine le plus spécialisé (choisi et défini par l'administrateur) et des données de provenance brutes (un fichier de log par exemple) pour pouvoir créer un premier modèle de domaine instancié appelé MDI (voir Figure 3.1 - partie C). Une fois ce premier MDI créé, d'autres MDIs peuvent être générés automatiquement pour créer un treillis de MDIs dont la borne supérieure est le modèle minimal instancié ($MDM_{instancié}$) et la borne inférieure est le modèle d'import ($MDI_{import}$). Ainsi, plusieurs niveaux d'interprétations sont créés, chacun parmi eux correspond à un MD donné et donc à une vue métier spécifique.

## 3.3 MDM : un Modèle de Domaine Minimal

L'une de nos premières questions est l'interopérabilité syntaxique entre plusieurs sources de provenance ayant des modèles hétérogènes. Pour répondre à ce besoin, nous proposons de définir un modèle minimal. Ce modèle définit un formalisme minimal indépendant des choix technologiques et des formats de représentation. Il doit proposer différentes sérialisations depuis et vers plusieurs formats de représentation. Ainsi, il devient possible d'avoir plusieurs sources de provenance dont chacune gère ses données avec son propre format de représentation.

La question est comment définir ce modèle et sur quelle base ? Quelle est sa structure et sa sémantique ? La définition d'un nouveau modèle qui peut être utilisé comme formalisme standard de provenance était pour nous une démarche à éviter pour les deux raisons suivantes :

1. la définition d'un tel formalisme demande un consensus entre des utilisateurs et des communautés ayant des besoins différents ce qui nécessite beaucoup de temps,

2. des travaux proposant des formalismes de provenance favorisant l'interopérabilité ainsi que des efforts standardisation existent déjà.

Le modèle OPM que nous avons présenté dans l'état de l'art (voir p.22) se présente comme un modèle de référence pour la communauté de workflows scientifiques. Ce modèle a été défini comme un standard pour l'échange et l'interopérabilité inter-communauté de provenance, ce qui correspond à nos exigences. Aussi, plusieurs systèmes proposent des fonctions d'export de provenance compatibles avec OPM (voir 2.3). Pour ces raisons, nous proposons de choisir OPM comme base de définition du MDM.

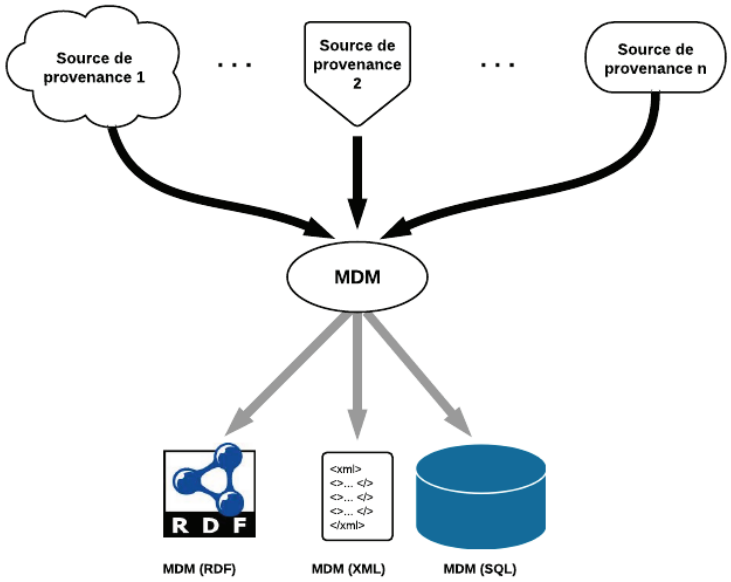


FIGURE 3.2 – Utilisation du MDM pour assurer l'interopérabilité syntaxique de la provenance

### 3.3.1 OPM comme base de définition du MDM

Dans les spécifications fonctionnelles du modèle OPM [Moreau *et al.*, 2008], les exigences annoncées sont les suivantes :

- permettre d'**échanger** des informations de provenance entre systèmes en utilisant une couche de compatibilité basée sur un **modèle de provenance partagé**,
- permettre aux développeurs de construire et de partager les outils qui fonctionnent sur un même modèle de provenance,
- définir **un modèle précis** et **indépendant des technologies d'implémentation**,
- fournir une représentation numérique de la provenance de tout type d'objet (même les objets qui ne sont pas produits par des systèmes informatiques),
- définir un ensemble de règles identifiant les inférences valides pouvant être effectuées sur les graphes de provenance.

OPM identifie aussi un ensemble de non-exigences qui correspondent aussi à nos exigences sur ce modèle minimal que nous souhaitons développer. En effet, **OPM n'est pas responsable** de :

- spécifier les représentations internes que les systèmes doivent adopter pour enregistrer et manipuler la provenance, **ces systèmes possèdent la liberté de choisir leurs représentations internes**,
- les sérialisations d'OPM et les implémentation en XML, RDF ou autres doivent être définies dans des documents dédiés et non pas dans les spécifications d'OPM,
- **OPM ne spécifie pas de protocoles pour stocker la provenance, ni de type de répertoires de stockage**,
- **OPM ne spécifie pas des protocoles pour interroger** la provenance.

Nous avons présenté dans l'état de l'art les trois composants de base d'OPM ainsi que les cinq relations de causalité qui peuvent les lier (voir Figure 2.3). OPM introduit le notion de « rôle » comme un label qui peut être ajouté aux arcs *used*, *wasGeneratedBy* et *wasControlledBy* pour distinguer la nature de la dépendance qui existe quand plusieurs de ces arcs sont connectés à un même processus. Un rôle possède donc une nature purement syntaxique et n'a un sens que dans le contexte du processus (ou/et de l'agent) auquel il est relié.

OPM définit aussi la notion de compte (account) pour présenter avec différents niveaux de détail le déroulement d'une opération. Ceci permet de modéliser plusieurs scénarios d'exécutions et de décrire par exemple, qu'à partir d'un artifact A, plusieurs chemins sont possibles pour obtenir le même artifact B.

Afin de pouvoir capitaliser sur les connaissances et les propriétés d'un domaine et de pouvoir modéliser des contraintes spécifiques, OPM introduit la notion de « profil » qui est une spécialisation d'un graphe OPM. Cependant, la multiplicité de profils peut causer des problèmes d'interopérabilité surtout si les règles d'extension et de conformité à OPM ne sont pas respectées. Quelques profils ont été proposé pour enrichir OPM comme le profil collections, le profil de signature et le profil de mapping vers Dublin Core [Moreau, 2010].

Pour nous, le cycle de vie d'un document peut se réduire dans sa forme la plus simple en une chaine d'événements que nous appelons chaine de provenance (ou provenance tout court). Cette chaine est composée d'un ensemble d'actions. Chacune d'entre elles est effectuée par un ou plusieurs acteurs.

Afin d'assurer une meilleure interopérabilité, nous devons garder une structure très simple et minimale pour le MDM. Pour cette raison, nous proposons **l'utilisation d'un sous-ensemble d'OPM**. En effet, nous n'avons pas besoin de représenter des exécutions différentes de processus ou d'actions. Ainsi, nous considérons uniquement trois type de connexions que nous utiliserons pour notre représentation minimale. Ce sont les arcs : (1) used (un process a utilisé un artifact), (2) wasGeneratedBy (un artifact a été généré par un process) et (3) wasControlledBy (un process a été contrôlé par un agent).

### 3.3.2 Construction du MDM

Nous créons notre MDM en héritant du modèle OPM les concepts et les propriétés qui nous intéressent. Nous créons notre concept *Action* en héritant du concept *Process*, le concept *Document* en héritant du concept *Artifact* et le concept *Actor* en héritant du concept *Agent*. Le MDM que nous proposons se résume donc en un graphe OPM réduit, défini par MDM (ActionId, DocumentId, ActorId, Valeur, Arc) avec :

- *ActionId* : un ensemble d'identifiants d'actions,
- *DocumentId* : un ensemble d'identifiants de documents,
- *ActorId* : un ensemble d'identifiants d'acteurs,
- *Valeur* : un ensemble de valeurs spécifiques à l'application,
- *Used* : ActionId × DocumentId ;
- *WasGeneratedBy* : DocumentId × ActionId ;
- *WasControlledBy* : ActionId × ActorId ;
- $Arc \subseteq Used \cup WasGeneratedBy \cup WasControlledBy$ ;

En ce basant sur ce formalisme, une instance du graphe de provenance est un 5-uplet. Ce graphe est définit par :

mdm = $< ActionId, DocumentId, ActorId, ValeurArc, valeurDe >$ où :

- valeurDe est de type ValeurDe où ValeurDe : Node → $\mathbb{P}$(Valeur),
- Node = ActionId ∪ DocumentId ∪ ActorId.

**Definition 1** (Modèle de Domaine Minimal (MDM))**.** *Un MDM est un modèle qui peut être formalisé dans une ontologie comme suit :*
- *un ensemble de classes disjointes CB : Action, Document, Actor,*
- *un ensemble de propriétés P : <documentId, actionId, actorId, applicationId, timestamp>,*
- *un ensemble de relations R : <hasDocumentId, hasActionId, hasActorId, hasApplicationId , hasTimeStamp, precededBy>,*
  *Chaque relation appartenant à R a un domaine dans CB et un co-domaine dans P comme suit :*
  - *hasDocumentId : Document → documentId,*
  - *hasActionId : Action → actionId,*
  - *hasActorId : Actor → actorId,*
  - *hasApplicationId : CB → applicationId,*
  - *hasTimestamp : Action → timestamp,*
  - *precededBy : Action → Action.*

Nous présentons sur la Figure 3.3 une modélisation du MDM sous forme d'une ontologie. Sur cette figure, nous avons utilisé le vocabulaire OPMV [1] qui fournit un vocabulaire conforme au modèle OPM. Pour cette modélisation, nous avons utilisé aussi des concepts de l'ontologie OPMO [2] qui est la spécification OWL d'OPM.

1. Open Provenance Model Vocabulary : http ://open-biomed.sourceforge.net/opmv/ns
2. www.openprovenance.org/model/opmo

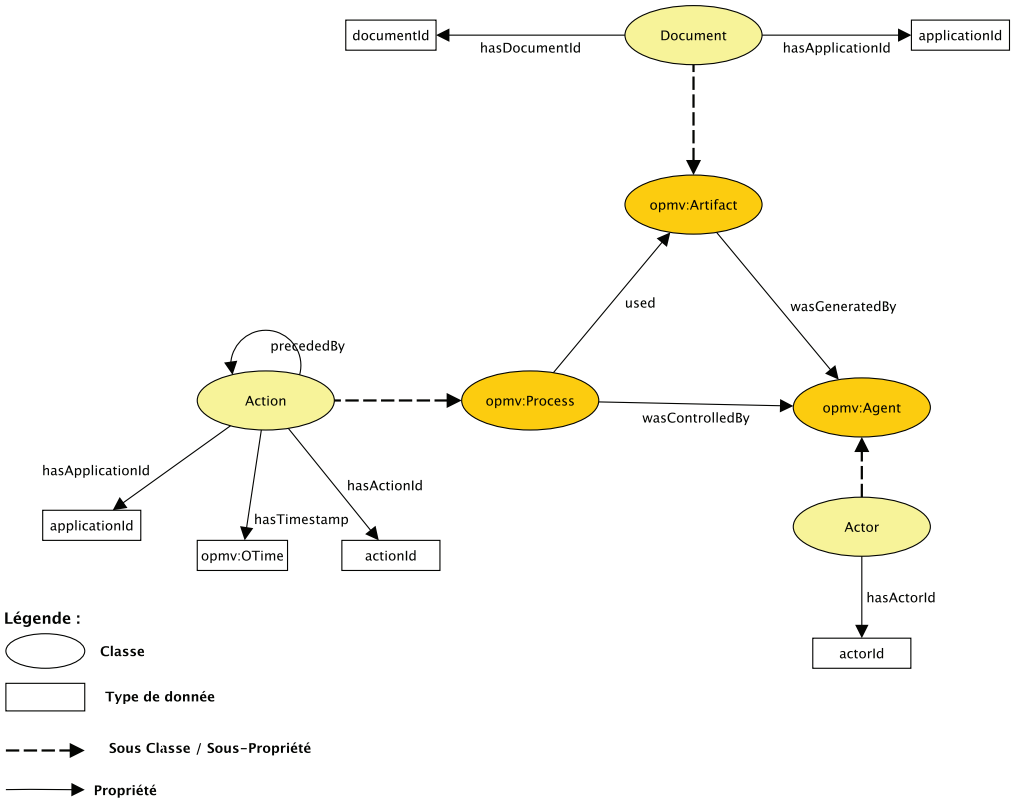


FIGURE 3.3 – Définition du modèle MDM en se basant sur OPM

Dans notre travail, nous allons utiliser le mot **artifact** pour désigner un objet de première classe appartenant à l'une des classes de base (CB) de notre MDM . Ainsi un artifact peut être un **Document**, un **Actor** ou une **Action**.

### 3.3.3 Synthèse

Nous avons présenté dans cette première partie comment construire notre modèle de domaine minimal (MDM). Pour assurer sa généricité, nous avons proposé d'utiliser un sous-ensemble du modèle OPM. Le $MDM$ proposé assure de ramener les données de provenance correspondantes à plusieurs types de modèles sources à un niveau d'interprétation commun et minimal. Étant donné que ce modèle est abstrait et ne dépend pas d'un format de représentation unique, l'interopérabilité syntaxique entre des sources de provenance hétérogènes est assurée.

## 3.4 Les modèles de domaine

Le modèle OPM a introduit la notion de profil permettant de spécialiser les graphes OPM pour traiter un problème spécifique ou pour modéliser de nouveaux besoins inexistants

dans le modèle OPM de base. Cependant, le niveau de liberté offert pour la définitions de nouveaux profils peut engendrer des problèmes d'interprétation et d'interopérabilité [Simmhan, 2009]. Pour répondre aux besoins d'extensibilité et de définition de modèles encapsulant des concepts et des propriétés métier, nous proposons de modéliser les spécificités d'un domaine donné dans un modèle que nous appelons modèle de domaine ($MD$). Afin d'éviter les problèmes rencontrés sur l'extension des profils OPM, nous proposons dans notre framework d'utiliser un mécanisme basé sur la spécialisation du $MDM$. Cette spécialisation permet de créer de nouveaux $MDs$ en gardant la conformité avec le $MDM$.

### 3.4.1 Formalisation des modèles de domaine

Un $MD$ regroupe les concepts et les propriétés d'un métier et décrit ses spécificités. Il s'agit d'une spécialisation du $MDM$ qui consiste à définir de nouveaux concepts, propriétés et relations qui étendent le modèle $MDM$. Il est défini comme suit :

**Definition 2** (Modèle de domaine MD)**.** *Ce modèle est une spécialisation du MDM (et nous notons $MDM \leq_{spec} MD_i$). Il est défini par :*
- *un ensemble de sous-classes (SC) qui héritent des classes de base (CB) du MDM ,*
- *un ensemble de types D,*
- *un ensemble de propriétés, chaque propriété a un domaine dans SC et un co-domaine dans D,*
- *un ensemble de relations inter-classes $R \in (CB \cup SC) \times (CB \cup SC)$.*

Pour créer un nouveau $MD$, il suffit de spécialiser au moins une classe parmi les classes CB du $MDM$. Il est permis aussi de concevoir son $MD$ par l'union de plusieurs $MDs$ à condition de vérifier que chacun entre eux spécialise un concept qui n'est pas déjà spécialisé dans un autre $MD$ et que cette union de $MDs$ ne génère aucune incompatibilité sémantique.

En définissant les modèles par ce principe de spécialisation, nous favorisons l'interopérabilité entre modèles et nous réduisons les ambiguïtés qui peuvent exister. Le modèle présenté ci-dessous (voir Figure 3.4) détaille les concepts et les propriétés d'un modèle de domaine pour l'archivage de documents à valeur probante. Nous avons réalisé ce modèle pour Novapost à partir de leur exigences métier sur la provenance des documents électroniques qu'ils traitent et archivent. Cette modélisation se base sur notre connaissance du métier d'archivage légal et de ses spécificités. Dans ce modèle, nous spécialisons les concepts ***Document***, ***Action*** et ***Actor*** du $MDM$.

Sur ce modèle (que nous notons $MD_{Novapost}$), nous introduisons le concept de document à valeur probante ***ProbativeValueDocument*** qui est la spécialisation du concept *Document*. Ce concept contient des méta-données sur l'intégrité comme l'empreinte SHA-256 du document, sa signature XAdES [3], son emplacement d'archivage sécurisé, ou encore sa trace XML. L'empreinte numérique peut être utilisée pour vérifier l'intégrité du document,

3. XML Advanced Electronic Signature. http ://www.w3.org/TR/XAdES

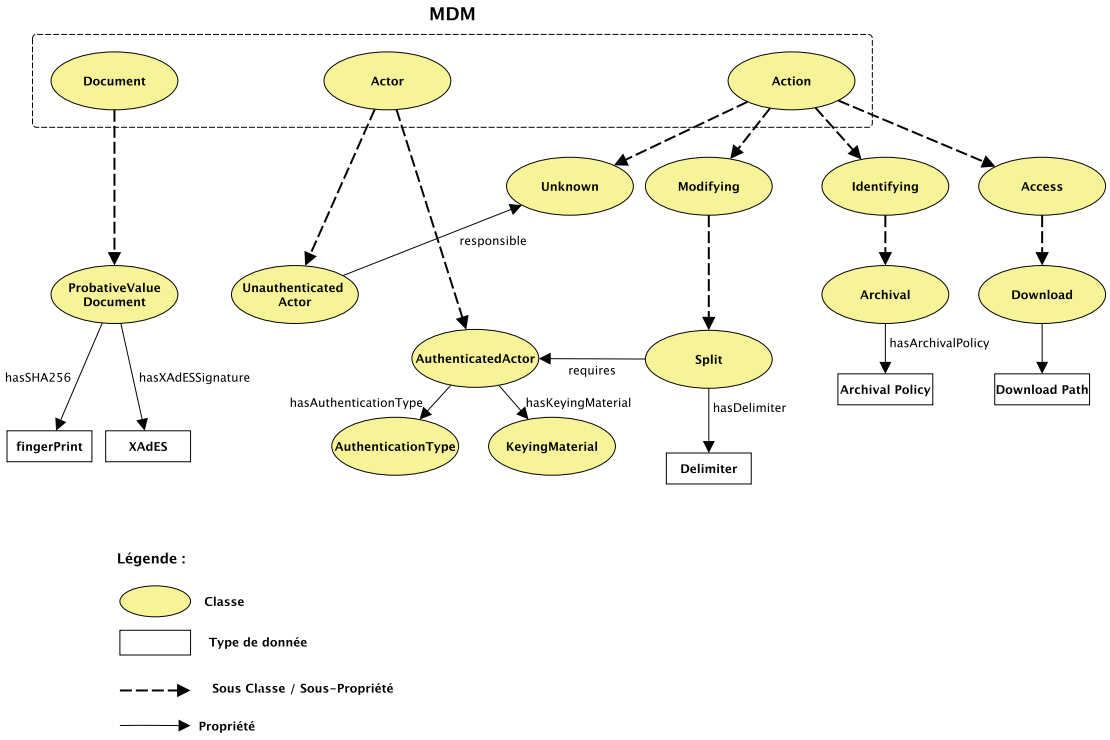


FIGURE 3.4 – Spécialisation du MDM pour générer un modèle métier ($MD_{Novapost}$)

la signature permet d'assurer la non répudiation et l'attribution d'actions aux entités qui l'effectuent. Les informations sur l'emplacement d'archivage permettent de vérifier si le document a été archivé dans des emplacements sécurisés respectant la politique d'archivage associée au document. La trace XML est un format de trace propriétaire à Novapost qui contient l'historique des actions effectuées sur le document.

Par spécialisation du concept acteur *Actor* du MDM, nous définissons le concept d'acteur authentifié ***AuthenticatedActor*** et d'acteur non authentifié ***UnauthenticatedActor***. Le besoin est de pouvoir identifier les entités qui ont effectué des actions sur les documents. En effet, pour certain types d'actions, les acteurs doivent être identifiés et authentifiés. D'autres types d'actions ne nécessitent pas ce type de contrainte. Les acteurs authentifiés doivent être associés à un type d'authentification. Différents types d'authentification existent et sont considérés comme des instances du concept ***AuthenticationType***. Pour les Webs service par exemple, on trouve plusieurs types d'authentification comme Http Basic, Http Digest, WSSW UserName Token ou OAuth.

Finalement, nous spécialisons le concept action du MDM pour définir quatre nouveaux concepts : un premier concept ***Modifying*** définissant un type d'action qui modifie le contenu du document, un deuxième concept d'action ***Identifying*** qui est une catégorie d'actions identifiantes qui consiste à créer un nouvel identifiant pour un document sur lequel elle est appliquée. Le troisième concept est ***Access*** qui définit les actions d'accès

aux documents comme la lecture ou le téléchargement. Le dernier concept d'action que nous définissons est ***Unknown*** qui définit une catégorie d'actions dont les acteurs ne sont pas identifiées ou authentifiés et dont les spécificités ne sont pas connues.

Ces catégories d'actions sont ensuite spécialisées. Nous définissons le concept ***Split*** qui consiste à découper un document selon un délimiteur prédéfini pour créer un ou plusieurs documents. Ce concept est une spécialisation du concept ***Modifying***. Le concept ***Archival*** est une spécialisation du concept ***Identifying*** qui consiste à archiver un document et à lui associer une politique, un identifiant et un emplacement d'archivage. Le concept ***Download*** est une spécialisation du concept ***Access*** qui consiste à télécharger le document en question.

De cette manière, nous modélisons les concepts et les propriétés associées au métier de l'archivage légal. Cette modélisation nous permet de décrire avec une granularité fine l'information de provenance dans ce cadre métier.

### 3.4.2 Les modèles de domaine instanciés

Les modèles de domaine instanciés ($MDI$) sont des instances des modèles de domaine. En utilisant une analogie avec le paradigme objet, les $MDs$ sont les classes et les $MDIs$ sont les objets. Concrètement, un MDI contient des informations sur la trace d'exécution des processus ou sur le déroulement d'un ensemble d'actions. Cette trace est annotée avec les concepts et les propriétés d'un $MD$ et contient ainsi les informations sur les actions, leurs entrées, leurs sorties ainsi que les entités qui les effectuent. Ces $MDIs$ peuvent être crées à partir d'une ou plusieurs sources de provenance qui peuvent être hétérogènes.

**Definition 3** (MDI : Modèle de Domaine Instancié)**.** *Un $MDI$ est l'instanciation d'un $MD$. Il contient des données de provenance annotées avec des concepts et des propriétés métier.*
*Un $MDI$ est associé à un seul modèle de domaine et à une (ou plusieurs) sources de provenance. Il est généré par l'intermédiaire d'une fonction d'import ou fourni nativement quand cela est possible.*

Pour pouvoir interpréter des données de provenance d'un $MDI_{source}$ (associé à un $MD_{source}$) en utilisant un autre modèle de domaine $MD_{cible}$, nous avons besoin de créer un nouveau $MDI_{cible}$ correspondant au $MD_{cible}$. Notre approche permet la génération de nouveaux $MDIs$ par application d'un ensemble de règles de généralisation. Cette opération de généralisation peut être vue comme une nouvelle classification des instances du $MDI_{source}$. En effectuant cette opération sur le même $MDI_{source}$ vers plusieurs $MD_{cible}$, nous générons des vues métiers sur les mêmes données de provenance. Ce processus de généralisation sera détaillé dans la partie 3.4.4.

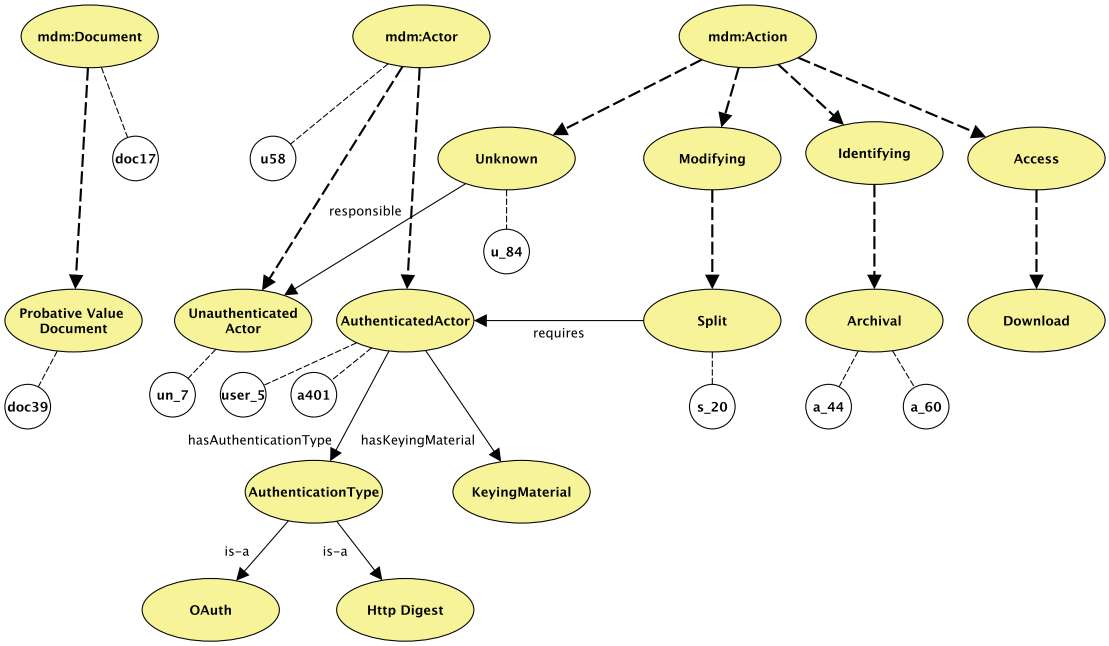


FIGURE 3.5 – Un $MDI$ correspondant à une instanciation du $MD_{Novapost}$

### 3.4.3 La fonction d'import de provenance

Si les applications et les services de logs ne génèrent pas directement des données conformes au $MDM$ (ou une de ses spécialisations), une fonction d'import doit être produite. D'un point de vue fonctionnel, la fonction d'import a pour rôle la génération du $MDI$ le plus spécialisé noté $MDI_{import}$. Elle prend en entrée la modèle de domaine le plus spécialisé appelé $MD_{import}$ et des données de provenance provenant d'une ou plusieurs sources (logs, formats spécifiques...) pour générer le $MDI_{import}$. Elle doit être écrite par un expert de domaine qui connait les spécificités métier de l'application (comme un administrateur qui connait les détails métier de son application et les types de requêtes de provenance qui seront posées).

Dans notre approche, une seule fonction d'import correspondant au $MD_{import}$ suffit pour un ensemble de sources de provenance données. Une fois le modèle de domaine le plus spécialisé instancié, la question qui se pose est comment instancier les autres modèles de domaine et créer de nouveaux $MDIs$ sans avoir à écrire une nouvelle fonction d'import pour chaque $MD$.

### 3.4.4 Généralisation des modèles de domaine instanciés

La généralisation d'un modèle de domaine instancié est une opération qui consiste à générer à partir d'un $MDI_{source}$ un nouvel $MDI_{cible}$. Effectivement, $MDI_{source}$ correspond à un $MD_{source}$ et $MDI_{cible}$ correspond à un $MD_{cible}$. La condition nécessaire et suffisante pour que la généralisation soit possible est : $MD_{cible} \leq_{spec} MD_{source}$ Ainsi, il devient

possible de générer de nouveaux $MDIs$ sans avoir besoin d'écrire une nouvelle fonction d'import spécifique pour chaque $MD_{cible}$.

Dans notre approche, nous avons au début un $MDI$ unique qui est le $MDI_{import}$. Donc, la généralisation ne peut être effectuée au début que sur ce $MDI$. Ensuite, elle peut être itérée sur les $MDIs$ nouvellement crées pour générer de nouveaux $MDIs$.

Par application de la généralisation des $MDIs$, nous assurant une compatibilité sémantique entre les différents $MDIs$ manipulés. Nous définissons la relation de généralisation pour un modèle de domaine instancié et nous la notons $\leq_{gen} : MDI \rightarrow MDI$.

$MDI_{cible} \leq_{gen} MDI_{source}$ ($MDI_{cible}$ est une généralisation de $MDI_{source}$) si est seulement si :

- Règle 1 : pour chaque concept du $MD_{source}$, toutes les instances de $MDI_{source}$ deviennent des instances du concept parent défini dans $MD_{cible}$ (toutes les propriétés et les relations qui ne sont pas définies dans le modèle de domaine cible $MD_{cible}$ pour ce concept seront supprimées),
- Règle 2 : pour chaque relation entre les concepts du $MD_{source}$ : toute relation entre deux concepts qui n'existent pas entre les deux concepts parents dans le domaine cible ($MD_{cible}$) sera supprimée.

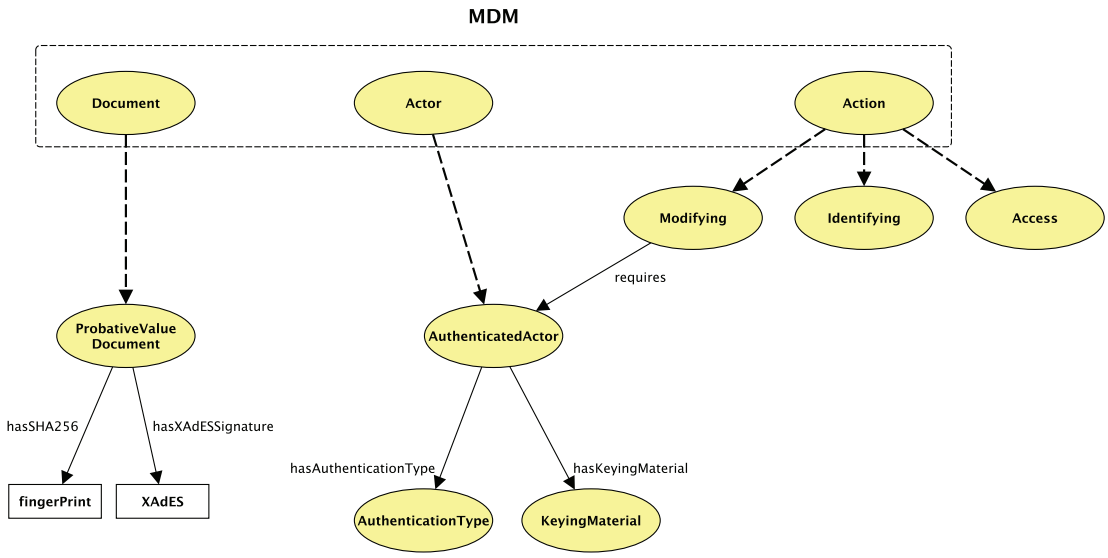


FIGURE 3.6 – Modèle de domaine cible $MD_{gen}$ pour la généralisation

Par exemple, si nous considérons comme $MDI_{source}$ à généraliser le $MDI_{Novapost}$ instance du $MD_{Novapost}$ (voir Figure 3.5) et comme $MD_{cible}$ un modèle où on ne souhaite pas divulguer tous les détails sur les opérations effectuées au sein du processus d'archivage Novapost (noté $MD_{gen}$), la généralisation nous permettra d'obtenir un $MDI_{gen}$ qui cache

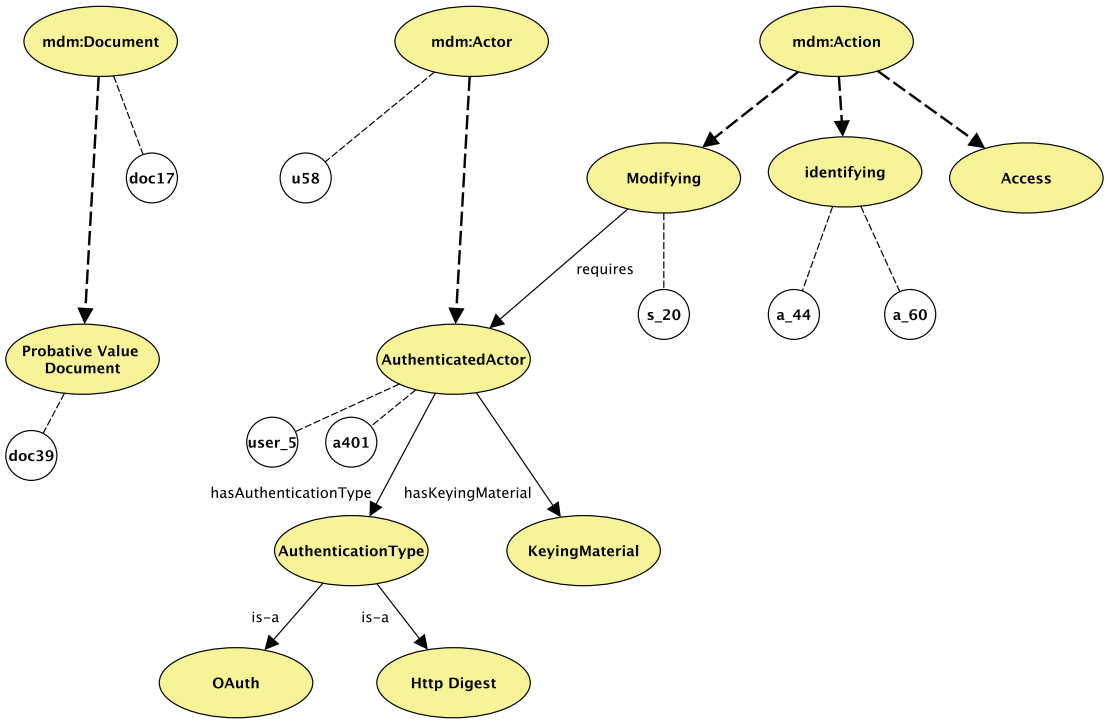


FIGURE 3.7 – Généralisation d'un $MDI_{Novapost}$ vers le modèle $MD_{gen}$

des détails métier mais qui permet quand même d'exprimer des informations sur l'intégrité des documents et sur l'identité des acteurs. Le résultat de cette généralisation est le $MDI_{gen}$ généré automatiquement par les règles de généralisation (voir Figure 3.7).

Par application de la règle de généralisation numéro 1 :

- **s_20** instance du concept *Split* devient instance de son concept parent *Modifying*,
- **s_44** et **a_60** instances du concept *Archival* deviennent instances du concept parent *Identifying*,
- **un_7** instance du concept *UnautheticatedActor* devient instance du concept parent *Actor* et **u_84** devient instance du concept parent *Action*.

Par application de la règle de généralisation numéro 2 :

- la relation *responsible* qui existait entre l'acteur **un_7** et l'action **a_84** instances respectives des concepts *UnautheticatedActor* et *Unknown* est supprimée et cette relation ne figurera pas sur le $MDI_{gen}$.

### 3.4.5 Génération d'un treillis de modèles de domaine instanciés

En partant du $MDI_{import}$ et en appliquant la relation de généralisation sur plusieurs modèles de domaine cibles $MD_{cible}$ nous générons un treillis de MDIs. Ce treillis possède $MDI_{import}$ comme borne inférieure et $MDM_{instancié}$ comme borne supérieure. Nous obtenons avec une fonction d'import unique et par application des règles de généralisation

autant de $MDI_{cible}$ que de $MD_{cible}$ souhaités. La seule condition est que ces $MDcible$ soient définis selon notre approche de modélisation basée sur la spécialisation du $MDM$. Ces $MDIs$ forment ainsi un treillis et sont considérées comme des vues métier sur les mêmes données de provenance.

L'existence du $MDM$ comme borne supérieure du treillis nous assure une interopérabilité syntaxique et un niveau d'interprétation minimal de n'importe quel $MDI$ en dessous. La définition des $MDs$ par spécialisation du $MDM$ assure une compatibilité sémantique entre les modèles de domaine et par conséquent entre les instances. Cette propriété nous permettra ultérieurement de faire la fusion entre plusieurs $MDIs$ et de corréler plusieurs identifiants distribués référençant le même document.

Par exemple, pour notre modèle de domaine d'archivage légal chez Novapost, nous avons un $MD_{Novapost}$=$MD_{import}$. Il décrit de façon détaillée les spécificités des actions effectuées sur les documents lors des processus d'archivage, les types de documents et leurs propriétés ainsi que les entités qui les effectuent. Pour un utilisateur du service d'archivage, il se pose des questions sur la provenance de ses documents. Ces questions sont différentes de celles posées par un auditeur qui va effectuer un audit sur le service d'archivage de Novapost. D'un autre coté, les questions de provenance qui intéressent l'administrateur du service d'archivage Novapost sont différentes de celles qui intéressent les deux acteurs précédents (utilisateur et auditeur du service).

Pour chacun de ces trois acteurs (utilisateur du service d'archivage, auditeur du service, administrateur du service), nous avons besoin de définir des modèles de domaine qui correspondent aux questions de provenance qu'ils souhaitent poser. Pour un utilisateur du service d'archivage, il se pose des questions de provenance basiques. Il s'interroge sur l'authenticité de ses documents et sur le fait que les acteurs qui les ont manipulés sont authentifiés. Pour modéliser ces besoins de provenance, nous spécialisons le $MDM$ pour créer le modèle de domaine utilisateur $MD_{User}$ (voir Figure 3.8).

Pour l'auditeur du service d'archivage, il se focalise sur l'utilisation de la provenance comme preuve de conformité légale. Il s'intéresse aux types d'actions effectuées sur le document et si les acteurs qui les ont réalisé étaient autorisés à le faire au moment où ils les ont réalisé. Un auditeur a aussi le droit d'accéder en clair au contenu du document. Les questions que cet acteur se pose sont plutôt focalisées sur l'utilisation de la provenance comme preuve légale. Pour ces raisons, son $MD$ ne s'intéresse qu'aux catégories des actions et non pas aux détails métier unitaires au niveau de chaque action. l'auditeur a aussi besoin d'avoir des informations sur l'identité des entités qui participent au traitement des documents et leurs informations d'authentification. En spécialisant le modèle $MD_{User}$, nous créons le $MD_{Auditor}$ (voir Figure 3.9).

Nous définissons ensuite le modèle de domaine de l'administrateur du service d'archivage $MD_{Administrator}$ en spécialisant le $MDM$ (voir Figure 3.10). En effet, l'administrateur de service doit connaitre toutes les spécificités du processus de gestion documentaire au sein de l'entreprise, l'identité des entités qui les effectuent et leurs autorisations. Cependant,

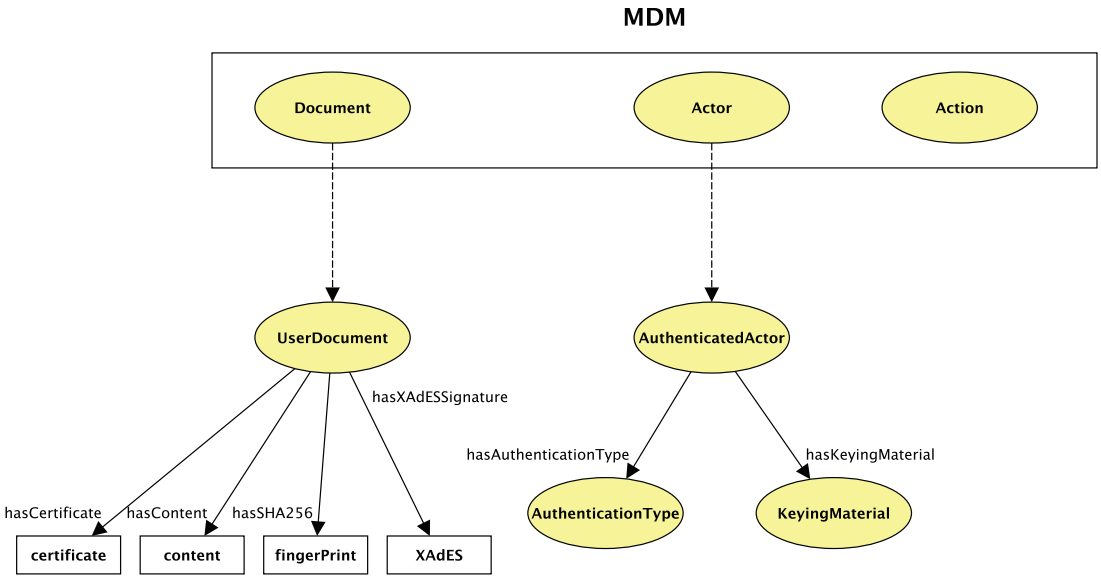


FIGURE 3.8 – Modèle de domaine pour un utilisateur du service d'archivage ($MD_{User}$)

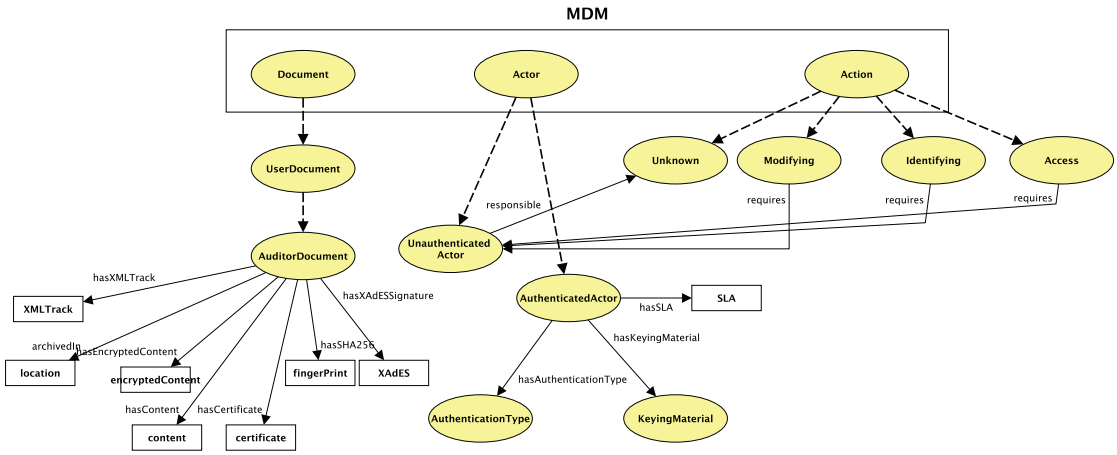


FIGURE 3.9 – Modèle de domaine pour un auditeur du service d'archivage($MD_{Auditor}$)

dans le cadre de l'archivage légal, la confidentialité est une obligation légale : les documents doivent être cryptés ou anonymisées. Ainsi, $MD_{Administrator}$ doit intégrer des informations assurant qu'il s'agit du contenu crypté ou anonymisé et non pas du contenu en clair. Il s'agit de la relation hasEncryptedContent définie par : hasEncryptedContent : ProbativeValueDocument → encryptedContent. Aussi, les acteurs authentifiés sont associés à un SLA pour pouvoir vérifier s'ils ont bien respecté les différentes clauses d'un contrat de service.

La fonction d'import permet d'instancier le $MD_{import}$ pour créer le $MDI_{import}$. Une fois que des modèles de domaine (spécialisations du $MDM$ et plus génériques que le $MD_{import}$ sont définis), nous pouvons utiliser les règles de généralisation pour créer les nouveaux

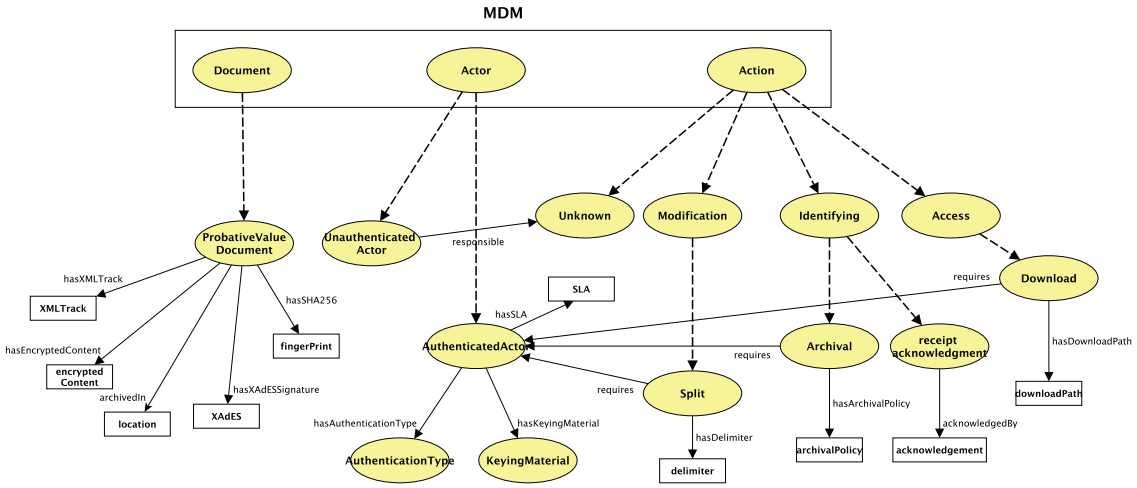


FIGURE 3.10 – Modèle de domaine pour un administrateur du service d'archivage ($MD_{Administrator}$)

$MDIs$. Pour notre exemple, nous pouvons créer automatiquement, à partir de notre $MDI_{Novapost}$ trois nouveaux modèles instanciés $MDI_{Auditor}$, $MDI_{Administrator}$ et $MDI_{User}$ qui correspondent respectivement aux modèles de domaine $MD_{Auditor}$, $MD_{Administrator}$ et $MD_{User}$ et qui peuvent être organisés comme un treillis (voir Figure 3.11).

En effectuant la gestion des modèles et des instances de provenance avec notre approche que nous venons de décrire, nous répondons à la deuxième partie de la question numéro 2 de notre problématique de recherche (présentée dans notre introduction générale) sur l'interopérabilité sémantique des modèles de provenance.

## 3.5 Gestion de multiples sources de provenance

### 3.5.1 Introduction

Dans la partie précédente, nous avons présenté notre approche et son application dans le cas où une seule source de provenance existe. En réalité, les sources de provenance sont multiples. Un document évolue à travers plusieurs systèmes le long de son cycle de vie. La trace des opérations effectuées est ainsi distribuée sur de multiples sources de provenance. Aussi, les identifiants des documents changent d'un système à un autre et des problématiques d'identification d'objets distribués interviennent et risquent de nuire à la complétude et la pertinence de la chaine de provenance.

Pour pouvoir exploiter, corréler et interroger des données de provenance provenant de plusieurs sources, nous proposons de les fusionner dans un seul $MDI$. Ce $MDI$ doit correspondre à un $MD$ qui doit unifier tous les $MDs$ des sources de provenance impliquées. Pour cela, il faut qu'il n'y ait aucun problème de compatibilité sémantique pour tous les $MDs$ à fusionner. Aussi, la corrélation entre plusieurs identifiants distribués doit être gérée

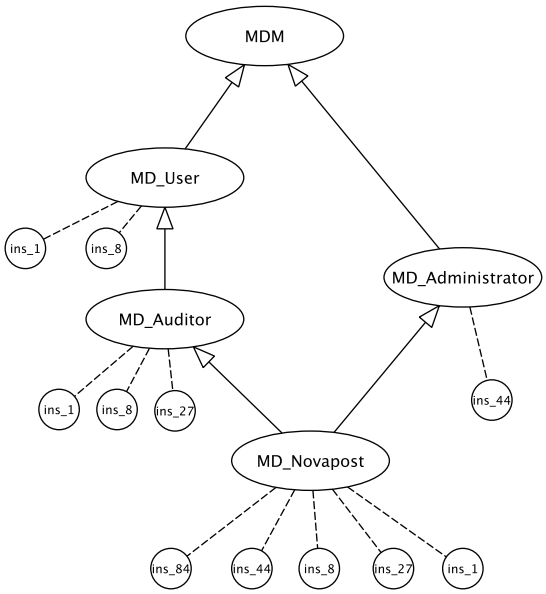


FIGURE 3.11 – Exemple d'un treillis de $MDIs$

pour assurer une meilleure lisibilité et une complétude de la chaine de provenance. Ce processus nous permettra de créer des liens entre les artifacts et d'inférer de nouvelles relations et propriétés.

Pour pouvoir fusionner des $MDIs$ correspondants à des $MDs$ différents, nous devons identifier un modèle de domaine de fusion ($MD_{fusion}$) sur lequel la fusion des différentes instances des $MDs$ en question est possible. Pour cette raison, ces $MDs$ doivent être compatibles. Cette compatibilité peut être assurée en utilisant des algorithmes d'alignement sémantique [Euzenat et Shvaiko, 2007] ou bien en se basant sur ce que nous avons défini précédemment, notamment la relation de généralisation définie en 3.4.4.

Nous allons détailler ce processus de fusion qui se compose d'une étape de fusion de $MDs$ puis d'une étape de fusion de $MDIs$.

### 3.5.2 Processus de fusion des MDs

La première étape dans le processus de fusion consiste à identifier un modèle $MD_{fusion}$ sur lequel la fusion de l'ensemble des modèles de domaines $\{MD_i, .., MD_n\}$ est possible. Pour pouvoir les fusionner, ces modèles doivent être compatibles. Nous définissons tout d'abord la notion de compatibilité sémantique entre modèles de domaine. Intuitivement, deux $MDs$ sont compatibles s'ils ne classent pas de deux manières différentes le même concept.

Formellement, deux modèles de domaine $MD_i$ et $MD_j$ sont compatibles si et seulement si :

$$\begin{cases} \exists MD_k \ tel \ que \ MD_i \leq_{spec} MD_k \ et \ MD_j \leq_{spec} MD_k \\ MD_k \ est \ minimal : \nexists MD_l \ tel \ que \ MD_l \leq_{spec} MD_k \ et \ MD_i \leq_{spec} MD_l \ et \ MD_j \leq_{spec} MD_l \end{cases} \tag{3.1}$$

Nous explicitions maintenant les étapes permettant d'effectuer ce processus (la fusion) pour les différents cas de figures qui peuvent se présenter : (1) les modèles de domaine associés aux $MDIs$ à fusionner sont totalement ordonnés par la relation $\leq_{spec}$, (2) les modèles de domaine associés aux $MDIs$ à fusionner ne sont pas totalement ordonnés par $\leq_{spec}$ et (3) les modèles de domaine associés aux $MDIs$ à fusionner ne sont pas compatibles.

#### 3.5.2.1 Cas où les modèles de domaine à fusionner sont totalement ordonnés

Nous considérons n sources de provenance générant n $MDIs$ $(MDI_1, MDI_2, ..., MDI_n)$ avec $(MD_1, MD_2, ..., MD_n)$ les modèles de domaine respectivement associés. Si ces $MDs$ sont totalement ordonnés en utilisant la relation de spécialisation entre $MDs$ $(MDM \leq_{spec} MD_i \leq_{spec} ... \leq_{spec} MD_n)$ alors ils sont compatibles sémantiquement. Des cas de compatibilités spéciaux existent :

- tous les $MDs$ en entrée sont identiques $(MD_i, MD_i, ..., MD_i)$, dans ce cas trivial $MD_{fusion} = MD_i$,
- les $MDs$ appartiennent à deux catégories différentes, le $MDM$ et un autre $MD$ $(MDM, MDM, ..., MD_i, MD_i)$, dans ce cas $MD_{fusion} = MD_i$. Nous avons ce cas de figure quand il s'agit par exemple d'un ensemble de sources qui génèrent la provenance dans un log http (qui correspond au modèle MDM) et dans un log applicatif (qui correspond à un modèle $MD_i$).

Pour illustrer, nous prenons l'exemple de fusion entre le modèle de domaine auditeur $MD_{Auditor}$ (voir Figure 3.9) et le modèle de domaine utilisateur $MD_{User}$ (voir Figure 3.8). Ces deux modèles sont des spécialisations du $MDM$. Ils sont ordonnés avec la relation de spécialisation et vérifient :

$$\begin{cases} MDM \leq_{spec} MD_{User} \\ MDM \leq_{spec} MD_{Auditor} \\ MD_{User} \leq_{spec} MD_{Auditor} \end{cases} \tag{3.2}$$

Dans ce cas, $MD_{Auditor}$ (voir Figure 3.9) est le modèle spécialisé minimal assurant la compatibilité pour $MD_{User}$ et $MD_{Auditor}$. Ainsi, $MD_{fusion}$=$MD_{Auditor}$.

#### 3.5.2.2 Cas où les modèles de domaine à fusionner sont compatibles mais pas totalement ordonnés

Dans le cas plus général où les $MDs$ ne sont pas totalement ordonnés avec la relation de spécialisation, $(MD_i \not\leq_{spec} ... \not\leq_{spec} MD_j)$, la fusion est néanmoins possible si les $MDs$ sont compatibles sémantiquement. Nous retrouvons ce cas de figure quand il s'agit par

exemple de deux logs applicatifs différents et complémentaires. Chacun de ces deux logs concerne un processus métier bien défini.

Pour illustrer, nous prenons le cas d'un processus de fusion entre les deux modèles de domaine $MD_{Admin1}$ et $MD_{Admin2}$ présentés ci-dessous (voir Figure 3.12). $MD_{Admin1}$ spécialise uniquement le concept *Document* du $MDM$ alors que $MD_{Admin2}$ spécialise les concepts *Actor* et *Action* du $MDM$.

Ces deux $MDs$ ne sont pas totalement ordonnés avec $\leq_{spec}$ et nous avons :

$$\begin{cases} MD_{Admin1} \not\leq_{spec} MD_{Admin2} \\ MD_{Admin2} \not\leq_{spec} MD_{Admin1} \end{cases} \tag{3.3}$$

Bien que $MD_{Admin1}$ et $MD_{Admin2}$ ne sont pas totalement ordonnées par $\leq_{spec}$, ils sont compatibles. En effet, nous avons :

$$\begin{cases} MD_{Admin1} \leq_{spec} MD_{Administrator} \\ MD_{Admin2} \leq_{spec} MD_{Administrator} \\ MD_{Administrator}\ est\ minimal. \end{cases} \tag{3.4}$$

Ainsi, nous avons $MD_{fusion}$=$MD_{Administrator}$ (voir Figure 3.10).

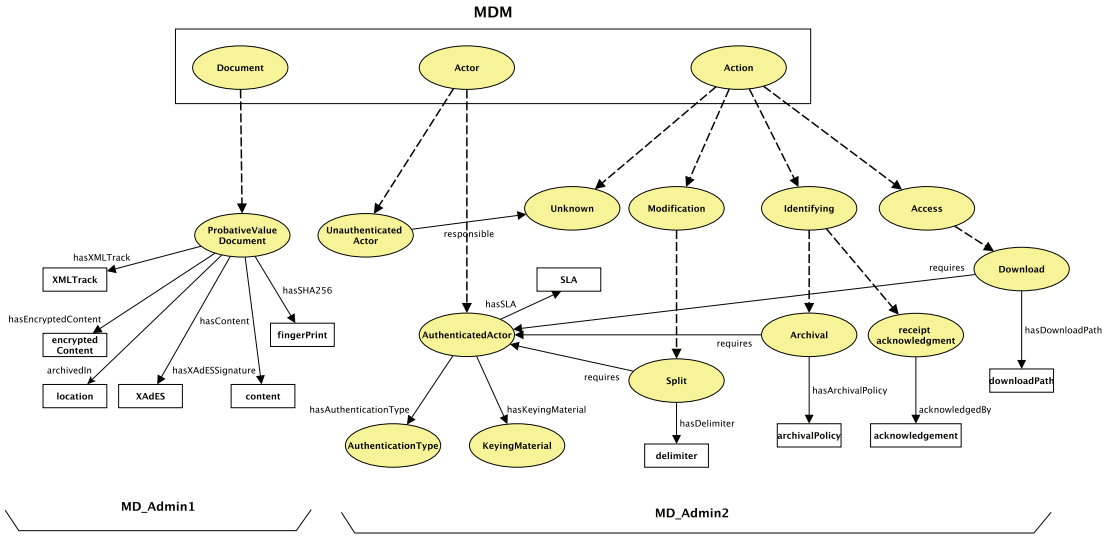


FIGURE 3.12 – Exemple de MDs pas totalement ordonnées et compatibles : $MD_{Admin1}$ et $MD_{Admin2}$

#### 3.5.2.3 Cas où les modèles de domaine à fusionner ne sont pas compatibles

Dans le cas où les $MDs$ à fusionner ne sont pas compatibles, nous pouvons nous ramener aux cas précédents par généralisation des $MDs$. Ce cas de figure est rencontré si les $MDs$ concernés spécialisent de manière différente les mêmes concepts. Une façon de régler ce problème est d'utiliser la généralisation du $MDI$ qui de fait généralise le $MD$ associé. Ainsi,

la non compatibilité va disparaitre et on va pouvoir à un des deux cas précédents. On est sûr de pouvoir y arriver, au pire en ramenant tous les $MDs$ au niveau du $MDM$ ! Cette approche a cependant deux limites. Tout d'abord, il n'y a pas une manière unique de régler le problème et ensuite la généralisation entraine une perte d'informations tant au niveau du $MD$ que du $MDI$.

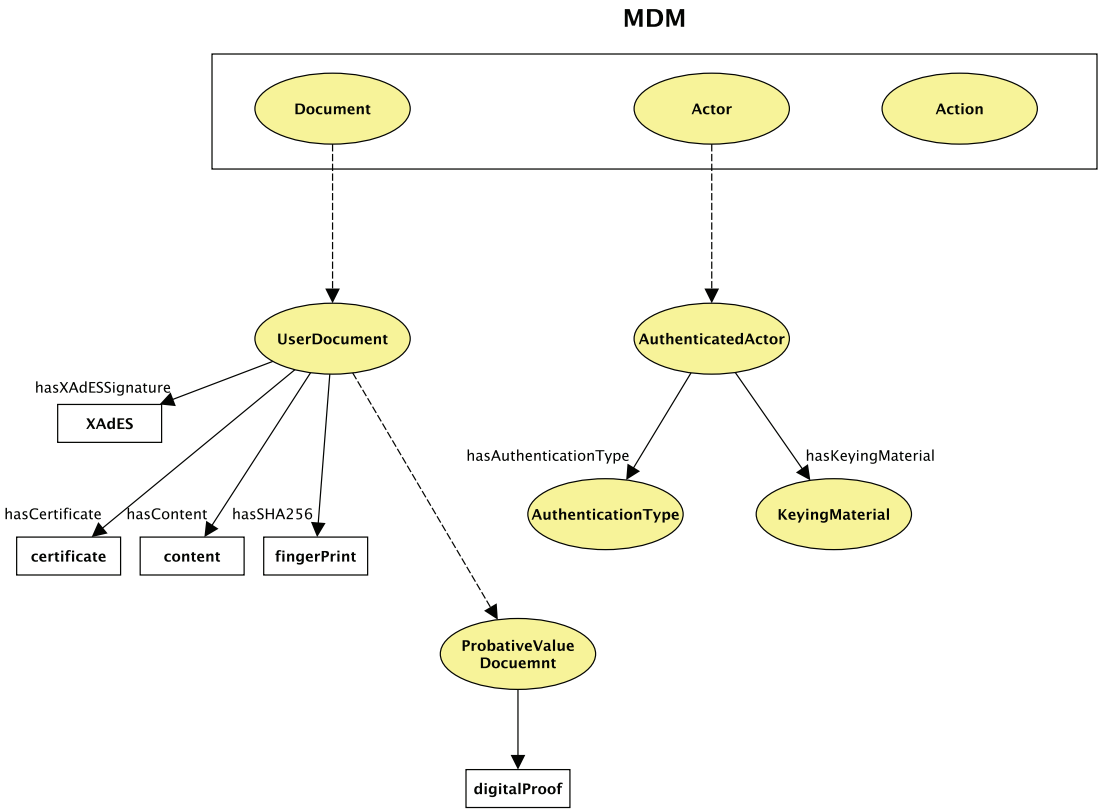


FIGURE 3.13 – Modèle de domaine $MD_{SpecificUser}$

Prenons par exemple le cas d'une fusion entre les modèles $MD_{SpecificUser}$ (voir Figure 3.13) et $MD_{Administrator}$ (voir Figure 3.10). Le modèle $MD_{SpecificUser}$ est similaire à $MD_{User}$ sauf qu'il spécialise encore le concept *UserDocument* pour créer le concept *ProbativeValueDocument*. Ces deux modèles ne sont pas compatibles car ils spécialisent de deux manières différentes le concept *Document*. Pour les rendre compatibles, nous proposons de généraliser les deux modèles en ramenant la branche document au seul concept *Document* du $MDM$. De cette manière, la compatibilité sur *Document* est assurée (voir Figure 3.14).

### 3.5.3 Gestion de la correspondance entre MDIs

Une fois la compatibilité au niveau des modèles obtenue, nous passons à la compatibilité au niveau des instances. En effet, le processus de fusion peut générer des incohérences entres les instances. Celles ci doivent être résolues par un administrateur ayant des connaissances

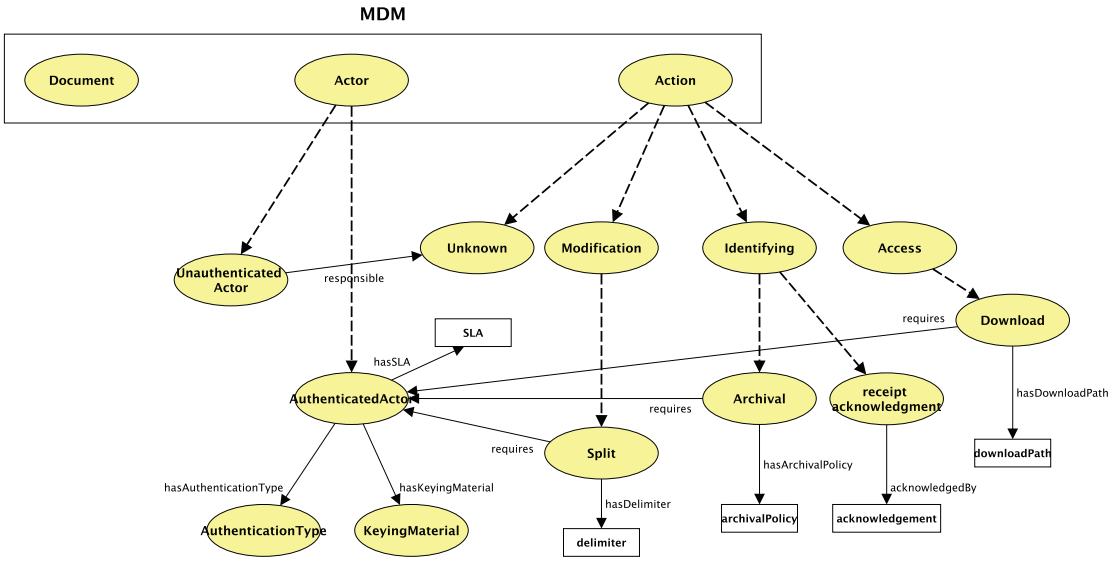


FIGURE 3.14 – Modèle de domaine de fusion $MD_{fusion}$ correspondant aux modèles $MD_{SpecificUser}$ et $MD_{Administrator}$

spécifiques (comme la personne ayant implémenté la fonction d'import par exemple). Le processus de fusion a pour but de relier des instances distribuées, de les corréler et de générer de nouvelles connaissances qui ne sont pas explicites si les sources de provenance sont gérées séparément.

Dans un contexte où les documents sont traités et échangés par des systèmes hétérogènes, les identifiants des documents et des acteurs le sont aussi. Ainsi, on se retrouve dans des cas où des identifiants différents référencent le même document. Pour résoudre cette problématique, nous proposons d'utiliser une **fonction de correspondance**. Cette fonction est fournie par un administrateur. Elle permet de relier des identifiants distribués référençant un même artifact lors du processus de fusion.

Dans notre contexte de travail chez Novapost, nous recevons des documents de paie des entreprise clientes. Ces documents sont reçus par un administrateur du service d'archivage pour les traiter et les archiver. Lors de la réception du document, l'administrateur acquitte la réception et renvoie à l'émetteur un identifiant. Cet identifiant est diffèrent de l'identifiant d'archive chez Novapost mais permet d'identifier le document chez Novapost depuis l'extérieur. Dans ce cadre là et afin d'assurer la pertinence et la lisibilité de la chaine de provenance des documents, nous avons besoin de gérer des informations de correspondance entre ces identifiants.

Cette fonction de correspondance peut être matérialisée sous plusieurs formes. Nous proposons d'utiliser une table de correspondance contenant un identifiant virtuel (TUUID : Track Universal Unique IDentifier) unique et partagé. Il référence pour chaque source l'identifiant local de l'artifact. Le tableau suivant (voir Table 3.15) illustre un exemple de

| $TUUID$ | $source\_id : artifact\_id$ |
|---|---|
| doc100 | r1:doc_89 , r2:cust012 |
| doc101 | r1:doc_28 , r2:doc_456,r3 :doc700 |

FIGURE 3.15 – Tableau de correspondance entre identifiants distribués

table de correspondance entre des identifiants distribués. Pour injecter cette information de correspondance dans les $MDIs$, nous proposons d'utiliser la propriété OWL *owl :sameAs* utilisée pour le mapping de ressources (dans notre cas, ces ressources peuvent être des documents, des actions ou des acteurs).

### 3.5.4 Processus de fusion des MDIs

Une fois le modèle de fusion ($MD_{fusion}$) identifié, la dernière étape consiste à fusionner les $MDIs$. Il s'agit donc d'instancier $MD_{fusion}$ pour créer $MDI_{fusion}$ à partir des différents $MDIs$ et des correspondances. Pour ce faire, ce processus réalise la projection de toutes les instances sur $MD_{fusion}$. Afin d'obtenir un $MDI_{fusion}$ complet, nous devons également intégrer les informations de correspondance entre les artifacts (toutes les relations $sameAs$ entre les artifacts).

Pour illustrer le processus de fusion entre $MDIs$, nous prenons l'exemple de deux $MDIs$ ($MDI1$ et $MDI2$) ayant comme modèles de domaine le $MDM$ et $MD_{Novapost}$ respectivement. $MDI1$ provient du client qui émet les documents pour Novapost et est sémantiquement pauvre alors que $MDI2$ provient de chez Novapost et contient des détails métier sur le processus de traitement et d'archivage de documents.

Pour notre exemple, les $MDs$ sont totalement ordonnées par $\leq_{spec}$ (cas numéro 1 pour la fusion des $MDs$) et nous avons $MD_{fusion}$=$MD_{Novapost}$. Étant donné que $MD_{fusion}$=$MD_{Novapost}$, nous avons $MDI_{fusion}$=$MDI1 \cup MDI2$. Il n'y a pas ici de projection des instances puisque les $MDs$ sont préservés. Il suffit ensuite d'intégrer les $sameAs$ pour enrichir $MDI_{fusion}$ et créer de nouveaux liens entre les artifacts.

Nous présentons $MDI_{fusion}$ sur la Figure 3.16. Cet $MDI$ présente de nouveaux liens entre les artifacts. Nous pouvons remarquer un lien entre le document identifié par "doc_89" chez l'entreprise émettrice et celui identifié par $cust012$ chez Novapost. Etant donné que ce dernier est un document à valeur probante, nous pouvons inférer les propriétés "fingerprint" et "XAdES" et connaitre leur valeur pour ce document. Également, nous créons par cette fusion un lien entre ce document (doc_89) et l'action de signature électronique effectuée par l'acteur authentifié "authA34".

Plus généralement, si $I1$ et $I2$ sont deux instances reliées par un $sameAs$, toutes les propriétés et relations de $I1$ vont être ajoutées à $I2$ et réciproquement toutes celles de $I2$ vont être ajoutées à $I1$.

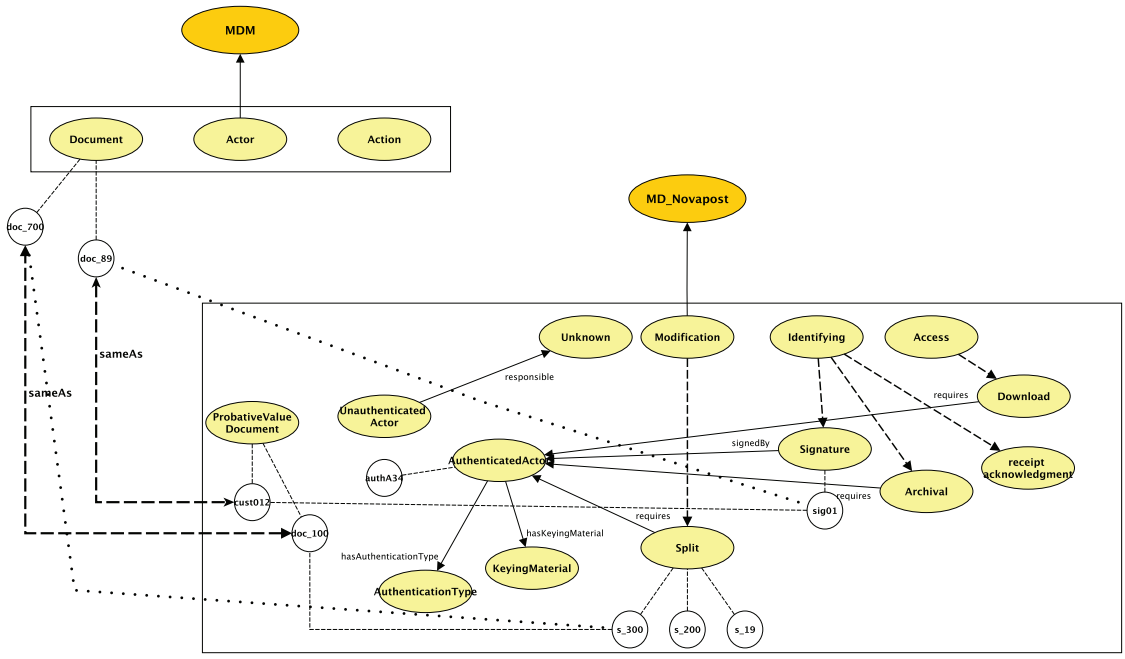


FIGURE 3.16 – Fusion entre un $MDI_{Novapost}$ et $MDM_{instancié}$

### 3.5.5 Synthèse

Nous avons détaillé dans les deux sections précédentes (section 3.4 et section 3.5 ) la modélisation des concepts et des propriétés métier avec notre approche et leur gestion. Pour ce faire, nous avons introduit la notion de modèles de domaine définis par spécialisation du MDM. Ces modèles peuvent être instanciés à partir de n'importe quelle source de provenance pour générer des modèles de domaine instanciés. Ces derniers peuvent être généralisés sur plusieurs modèles de domaines plus abstraits permettant ainsi la création d'un treillis de $MDIs$. Il devient ainsi possible d'avoir plusieurs vues métier sue les mêmes instances de provenance.

Dans la section 3.5, nous avons traité l'aspect multi-sources de la provenance et nous avons proposé un processus de fusion qui se déroule en deux étapes. La première fusionne les $MDs$ en un modèle de fusion alors que la seconde fusionne les instances. Nous avons présenté comment obtenir ce $MD_{fusion}$ pour les différents cas de compatibilité qui peuvent exister entre les $MDs$ en question. Nous avons traité ensuite la problématique d'identifiants multiples d'artifacts distribués sur plusieurs sources de provenance ainsi que le déroulement de la fusion au niveau des instances. La solution que nous avons proposé se base sur une fonction de gestion de correspondance qui permet de croiser les identifiants d'artifacts distribués à travers de multiples sources. Avec cette proposition de gestion de fusion, nous pouvons générer des propriétés et des relations qui ne sont pas explicites si les sources de provenance sont considérées séparément.

## 3.6 Conclusion

Dans ce chapitre, nous avons présenté notre approche de modélisation et de gestion de la provenance. Notre objectif est de pallier à l'hétérogénéité syntaxique et sémantique qui existe entre les différents modèles de provenance issus de sources hétérogènes. Pour ce faire, nous avons proposé une approche globale et cohérente pour la modélisation de la provenance basée sur les technologies du Web sémantique. Notre approche se base sur un modèle de domaine minimal assurant un niveau d'interprétation minimal et commun pour n'importe quelle source de provenance. Notre modèle MDM se base sur OPM qui est un modèle de provenance de référence. Il assure une interopérabilité au niveau syntaxique entre des modèles de provenance hétérogènes en se basant sur une structure simple et générique. Ainsi, il devient possible d'interpréter des données de provenance correspondants à des modèles et des sources hétérogènes avec un modèle commun ce qui assure l'interopérabilité syntaxique souhaitée.

Ensuite, l'idée consiste à pouvoir spécialiser ce modèle minimal en plusieurs modèles de domaine pour modéliser des concepts et des propriétés métier différentes. Ces modèles métiers sont formalisés sous formes d'ontologies en se basant sur les technologies du Web sémantique (notamment OWL). Ces modèles de domaine sont ensuite instanciés pour générer des modèles instanciés appelés $MDIs$. Nous définissons ensuite des règles de généralisation qui permettent de projeter les instances d'un modèle de domaine source ($MDI_{source}$) sur un modèle cible plus abstrait pour générer un $MDI_{cible}$. L'application de la généralisation d'un même $MDI_{source}$ sur des modèles de domaines cibles différentes permet de générer un treillis de $MDIs$. Ce treillis forme un ensemble de vues métier sur les données de provenance. Procédant ainsi, notre approche de modélisation basée sur la spécialisation des $MDs$, l'instanciation d'un $MDI_{import}$ riche que nous pouvons généraliser et la généralisation assure l'interopérabilité sémantique souhaitée entre les modèles de provenance.

Notre approche adresse aussi la problématique de gestion de multiples sources de provenance et propose un processus de fusion permettant d'unifier les $MDs$ et d'obtenir un seul $MD$ à interroger. Nous traitons ensuite la gestion de corrélation entre les identifiants d'artifacts distribués via une fonction de gestion de correspondance qui permet de lier les identifiants d'artifacts distribués sur plusieurs sources. Cela permet ensuite de générer de nouvelles informations impossibles à connaitre si les sources de provenance étaient considérées séparément.

Ainsi, nous répondons à la première et la deuxième question de notre problématique de recherche relatives à l'interopérabilité syntaxique et sémantique des modèles de provenance et l'intégration des concepts et des propriétés métier lors de la modélisation de la provenance.

La validation de ce travail a été faite à travers une publication dans une conférence internationale [Sakka *et al.*, 2012]. Dans le chapitre suivant, nous nous intéressons à la proposition d'architectures de PMSs qui mettent en œuvre notre approche de modélisation.

*Chapitre 4*

# Propositions d'architectures pour les systèmes de gestion de provenance

*"Il n'y a pas de plaisir plus complexe que celui de la pensée."*

Jorge Luis Borges, "L'immortel"

## Sommaire

## 4.1 Introduction

Après avoir présenté dans le chapitre précédent notre approche globale pour la modélisation de la provenance, nous nous intéressons dans ce chapitre à la conception des systèmes de gestion de provenance. Plus précisément, ce chapitre répond à la troisième et la quatrième question de notre thèse :« Quelle est l'architecture la mieux adaptée pour la conception d'un PMS qui couvre le cycle de vie de l'information de provenance ? » et « Quelle est l'architecture qui permet d'interroger les données de provenance dans un contexte distribué préservant l'autonomie des sources ? »

Notre approche détaillée dans le chapitre 3 a introduit les concepts et les composants nécessaires pour la collecte, la modélisation des spécificités métiers, la gestion de l'aspect distribué des sources et la génération des vues métier sur les données de provenance. Dans ce chapitre, nous nous intéressons à la mise en place d'architectures de PMSs mettant en œuvre notre approche de modélisation.

Deux types d'architectures sont proposées : la première est une architecture logique indépendante des choix technologiques et qui sert aux concepteurs et développeurs de PMSs comme une architecture de référence. Les différents modules de cette architecture peuvent être ensuite instanciés pour définir d'une façon technique détaillée l'architecture logicielle d'un PMS. Évidemment, chaque module de cette architecture assure une ou plusieurs fonctionnalités définies dans notre approche. La deuxième architecture est proposée dans un contexte distribué préservant l'autonomie des différentes sources de provenance et proposant une technique efficace pour le traitement des requêtes de provenance.

Le reste de ce chapitre est organisé comme suit. Dans la section 4.2, nous présentons les différents types de requêtes de provenance et les caractéristiques de chaque type. La section 4.3, détaille l'architecture logique de notre PMS et les fonctionnalités de chacun de ses composants. Ensuite, la section 4.4 s'intéresse à la proposition d'une version distribuée de PMS qui se base sur un médiateur ayant des capacités de résolution de requêtes. Nous présentons dans la section 4.5 l'algorithme de traitement de requêtes implémenté sur le médiateur. Nous détaillons à travers une série d'exemples comment s'effectue le traitement des requêtes avec notre algorithme. Chacune de ces requêtes possède des spécificités. Cela permet d'illustrer différents comportements du médiateur qui varient selon le type de la requête et la distribution des données. Enfin, nous concluons dans la section 4.6.

## 4.2 Requêtes de provenance : types et caractéristiques

Après notre étude bibliographique [Sahoo *et al.*, 2009], [Zhao *et al.*, 2011a] et [Zhao *et al.*, 2011b], nous avons identifié quatre catégories de requêtes de provenance :

1. Requête d'agrégation : elle regroupe un ensemble de valeurs en se basant sur des critères de provenance comme par exemple le nombre d'objets issus d'une opération ou la valeur minimale d'un attribut.

2. Requête forward : elle s'intéresse aux actions postérieures à une action donnée ou aux états postérieurs à un état donné d'un artifact. Ce type de requêtes permet d'identifier quelles sont les sorties qui ont été dérivées d'un artifact en entrée.
3. Requête backward : elle s'intéresse aux actions antérieures à une action donnée ou à l'ensemble d'états d'un artifacts antérieurs à un état donné. Ce type de requêtes permet d'identifier les entrées qui sont utilisées pour générer une sortie donnée.
4. Requête de chemin : elle s'intéresse à l'identification d'un chemin sur les données de provenance. Ce type de requêtes permet d'identifier le cycle de vie d'un artifact donné ou d'identifier les artifacts ayant un cycle de vie identique à un cycle de vie donné.

Selon notre vision, un PMS doit être capable d'exprimer ces quatre types de requêtes et d'y répondre. Nous présentons dans le tableau qui suit (voir Table 4.1) un ensemble de requêtes de provenance ainsi que leurs descriptions. Ces requêtes serviront par la suite à valider notre approche de modélisation et à tester le système développé.

| | **Requête de provenance** | **Description de la Requête et caractéristiques** |
|---|---|---|
| **Requête 1** | Quels sont les documents qui étaient traités par le module d'archivage du système Novapost ? | Requête backward avec contrainte sur l'acteur effectuant le traitement. |
| **Requête 2** | Que c'est il passé au document "doc-02-10-C" après son archivage ? | Requête forward identifiant les actions ultérieures à une action donnée sur un document donné. |
| **Requête 3** | Quel est le nombre de documents générés à partir du document "doc02-10-C" et qui sont archivés dans les coffres-fort électroniques chez le fournisseur de services "CDC" ? | Requête d'agrégation. |
| **Requête 4** | Est ce que la distribution du document "doc02-10-C" a été effectuée avec succès ? | Requête forward avec des contraintes métiers sur le succès de l'opération de distribution. Elle nécessite un modèle métier détaillant l'opération de distribution et ses critères d'échec et de réussite. Elle adresse aussi toutes les sources de provenance connues du système Novapost. |
| **Requête 5** | Quel est le cycle de vie complet du document "doc-02-2010-s2" pour l'administrateur Novapost ? | Requête de chemin. |
| **Requête 6** | Quel est le cycle de vie complet du document "doc-02-2010-s2" d'un point de vue auditeur ? | Cette requête est similaire à la requête précédente (requête 5). Cependant, il s'agit d'une vision auditeur du même cycle de vie. Elle se base donc sur un autre modèle métier, celui d'un auditeur. |

TABLE 4.1: Exemples et descriptions de requêtes de provenance.

## 4.3 Proposition d'une architecture logique pour un PMS

Afin de supporter la capture, le stockage, la gestion et l'interrogation de la provenance, nous présentons dans cette section l'architecture logique d'un PMS. Cette architecture (voir Figure 4.1) reprend tous les concepts introduits dans notre approche et les décline en un ensemble de modules proposant plusieurs fonctionnalités.

Le PMS proposé se base sur trois modules principaux offrant des fonctionnalités d'administration et de d'interrogation pour les administrateurs et des fonctionnalités d'interrogation pour les utilisateurs finaux. Ces modules sont : (1) un module administrateur pour la gestion de la provenance et ses modèles, (2) un module de traitement de requêtes et (3) un module de stockage. Notons qu'un administrateur du PMS est une personne ayant une vision globale sur les différentes sources avec lesquelles le PMS interagit, qui connait les détails métier de son application ainsi que les liens qui existent entre les modèles métiers. Par contre, un utilisateur d'un PMS est toute personne qui a besoin uniquement de fonctionnalités d'interrogation de provenance (i.e pouvoir faire des requêtes permettant de connaitre la provenance des artifacts, les nouveaux artifacts dérivés, la valeur de certaines propriétés relatives à la provenance ...) et donc n'as qu'une vision partielle des modeles métiers et des sources de provenance associées.

### 4.3.1 Module administrateur

Ce module propose un ensemble de fonctions de gestion des modèles et des instances de provenance (voir Figure 4.2). Il est accessible uniquement pour les administrateurs. Ces fonctions sont principalement l'import de sources de provenance, la gestion des sources de provenance, la gestion des $MDs$, la gestion des $MDIs$ et la gestion de la fusion des $MDIs$.

La première fonction, celle de l'import permet d'importer des données à partir d'une source spécifique (base de données, fichiers de logs...) ou d'un autre PMS vers le système de stockage. Cette fonction est l'implémentation de la fonction d'import que nous avons introduit dans notre approche (voir section 3.4.3). Vu que plusieurs types de systèmes de stockage peuvent être utilisés, son implémentation dépend également du système de stockage utilisé.

La deuxième fonction est la fonction de gestion des sources de provenance : un administrateur du système doit pouvoir déclarer et ajouter une ou plusieurs sources de provenance ou bien supprimer des sources existantes.

La troisième fonction est celle de gestion des modèles de domaine. Elle permet à l'administrateur de définir de nouveaux $MDs$ en spécialisant le $MDM$ ou un autre $MD$. Elle permet aussi de supprimer des $MDs$ qui existent déjà.

La quatrième fonction de ce module est la gestion des $MDIs$ et leur généralisation. La généralisation d'un $MDI$ permet de générer automatiquement un nouveau $MDI_{target}$ correspondant à un $MD_{target}$ à partir d'un $MDI_{source}$ correspondant à un $MD_{source}$. Dans

cette fonction, il s'agit de définir les règles permettant la génération des $MDIs_{target}$ à partir des $MDIs_{source}$.

La cinquième fonction est la fusion des $MDIs$. La fusion est une opération effectuée dans un contexte multi-sources : elle vise à intégrer plusieurs $MDIs$, de vérifier leur compatibilité sémantique et de les fusionner en un seul $MDI$. Cette fusion est effectuée par l'administrateur quand il s'agit de sources de provenance fortement liés. Elle permet de générer de nouvelles connaissances qui ne sont pas explicites si les sources de provenance sont traitées séparément. Ceci facilite l'interrogation de plusieurs $MDIs$ distribués selon un $MD$ donné. Cette fonction doit assurer la compatibilité sémantique des modèles à fusionner (comme détaillé dans notre approche en 3.5) ainsi que la compatibilité entre les instances pour éviter toute incohérence.

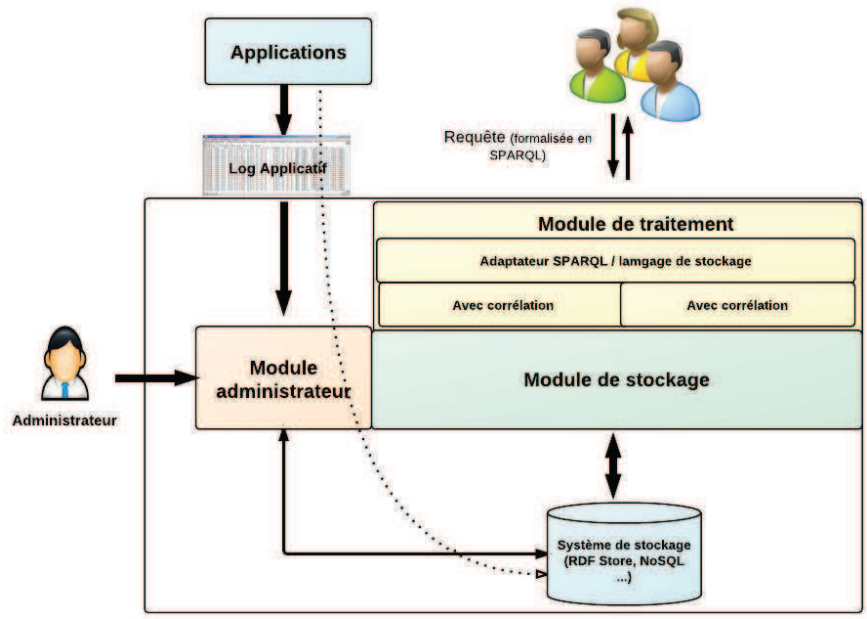


FIGURE 4.1 – Architecture logique proposée pour un PMS

### 4.3.2 Module de stockage

Ce module sert pour le stockage et la préservation à long terme de la provenance et des ses modèles. Il communique avec le module administrateur et le module de traitement de requêtes. Il peut utiliser n'importe quel système de stockage (une base de données relationnelle, un triple store RDF, une base de données NoSQL ...). Notons aussi que le système de stockage utilisé peut être centralisé ou distribué. Ce module peut proposer deux API :

– une première qui peut être utilisée par la fonction d'import pour stocker les données de provenance une fois que leur conversion depuis le modèle de données source vers le modèle de stockage est effectuée,

– une deuxième pour le stockage des logs applicatifs s'ils sont nativement formatés avec nos modèles de stockage.

### 4.3.3 Module de traitement de requêtes

Ce module traite les requêtes de provenance. Il prend en entrée une requête formalisée en SPARQL pour retourner des artifacts, des propriétés ou une chaine de provenance. Nous avons pris le choix d'utiliser SPARQL (ou un sous ensemble de SPARQL) comme langage pour la formalisation des requêtes car c'est le choix naturel associé à notre framework de modélisation. Cependant, nous n'imposons pas l'utilisation d'un RDF store comme système de stockage. Pour cette raison, ce module propose un adaptateur permettant de traduire les requêtes en entrée vers le langage de requête utilisé par le système de stockage.

En ce qui concerne le traitement d'une requête, il peut être effectué selon deux modes : 1) sans gestion de corrélation et 2) avec gestion de corrélation. Le premier mode correspond à un mode habituel de traitement où le requête est posée selon un schéma donné sur un ensemble de données. Ce traitement cherche à satisfaire les contraintes de la requête sur l'ensemble des données spécifiées.

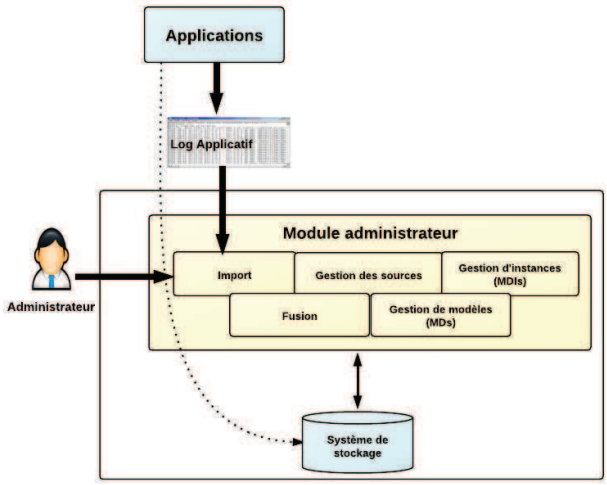


FIGURE 4.2 – Fonctions du module administrateur du PMS

Dans le deuxième mode (traitement de requête avec gestion de corrélation), il s'agit de prendre en compte les liens qui existent entre les différents artifacts pour rendre les réponses aux requêtes plus pertinentes. Donc l'idée ici est similaire à la fusion mais elle est appliquée dans un contexte de traitement de requête. En effet, la fusion est un processus lourd qui permet de fusionner des $MDIs$, de générer de nouvelles données et donc augmenter le volume des données stockées. Il est utilisé dans le cas où nous voulons accéder à ces données (générées par fusion) fréquemment. Contrairement à la fusion des $MDIs$, la gestion de corrélation n'augmente pas la taille de la base vu qu'elle effectue les inférences à la demande au moment du traitement de la requête. Il s'agit d'un processus adhoc qui s'effectue à la

volée lors du traitement de la requête et à la demande de l'utilisateur. Avec ce mode de traitement de requêtes, il devient possible de trouver des réponses plus pertinentes grâce à l'inférence sur les artifacts et leurs modèles. En effet, ce mode permet d'interpréter les *sameAs* qui lient les artifacts ce qui permet d'inférer de nouvelles propriétés et relations.

Pour illustrer, prenons l'exemple d'une requête qui cherche une propriété de provenance spécifique (*xmlTrack*) qui appartient au modèle $MD_{Novapost}$ pour le document identifié par doc_302. Les informations de provenance connues pour ce document correspondent au modèle $MDM$. Sans gestion de corrélation, la réponse à cette requête (voir Exemple de code 4.1) est vide puisque nous cherchons une propriété que le modèle $MDM$ ne possède pas. En supposant que l'administrateur a injecté des informations de correspondance établissant un lien *sameAs* entre l'artifact identifié par doc_302 et l'artifact identifié par doc_7802 et que cet artifact (doc_7802) correspond au modèle $MD_{Novapost}$, nous pouvons trouver une réponse à notre requête initiale. En effet, si nous interprétons le *sameAs* au moment du traitement de la requête, il devient possible d'inférer la valeur de la propriété *xmlTrack*. Un exemple plus détaillé sur le traitement de requête avec gestion de corrélation sera présenté dans le contexte distribué en 4.5.2.7

```
1 PREFIX nova: <http://track.fr/DM/probative.owl>
2 SELECT  ?track
3 FROM  <http://track.fr/applications/source1.rdf>
4 WHERE  {       ?documents nova:hasXMLTrack ?track .}
```

Exemple de code 4.1: Requête à traiter avec gestion de corrélation

### 4.3.4 Synthèse

Comme dans le travail effectué par Groth et al. [Groth *et al.*, 2009], nous avons proposé dans cette partie l'architecture logique d'un PMS, ses modules et leurs fonctionnalités. Leur architecture et la notre ont été proposées conjointement avec une approche complète pour la gestion de la provenance qui couvre ses étapes de collecte, stockage et interrogation. Cependant, le travail de Groth considère la construction d'un PMS à partir de zéro et définit aussi la manière avec laquelle les applications doivent structurer leurs données de provenance. Notre architecture, quant à elle considère l'utilisation des données de provenance existantes représentées dans n'importe quel type de structure. Sans être intrusive ou imposer des modifications au niveau des applications et systèmes sources, notre architecture offre aux applications les fonctionnalités assurant une gestion de bout en bout de leur provenance.

Dans le cadre de notre travail chez Novapost (dont la solution logicielle, ses services et ses modules seront détaillés dans le chapitre 5), le cycle de vie standard d'un document comprend des étapes de génération, de transfert, de traitement et d'archivage. Certaines de ces étapes sont effectuées par les workflows et les services Novapost, d'autres par des services tiers (comme les tiers archiveur) et d'autres par les SI clients (entreprises émettrices de documents). Dans une vision centralisée, l'architecture que nous proposons pourrait être déployée chez Novapost comme illustré dans la Figure 4.3.

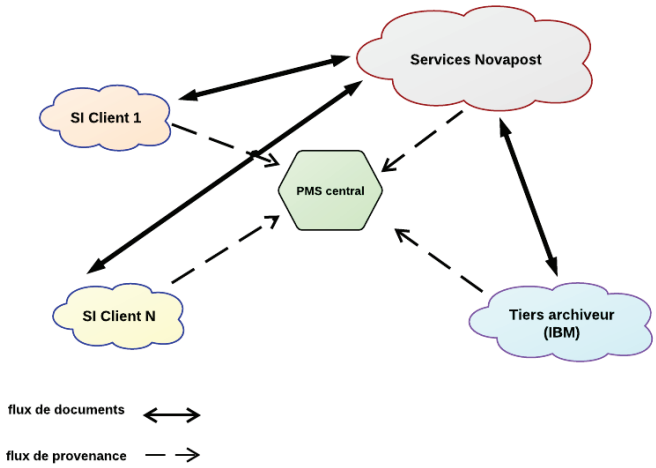


FIGURE 4.3 – Vue sur les interactions entre les systèmes d'une architecture PMS centralisée

Une architecture basée sur une gestion centralisée de la provenance permet de collecter toutes les données et d'interroger le module de stockage centralisé. Cependant, elle introduit de nouvelles contraintes sur la confidentialité des données de provenance parce que tous les modèles et les données seront hébergés dans un emplacement unique, le PMS central. Aussi, les questions de scalablité doivent être traitées en amont pour cette architecture qui doit supporter une nombre important de sources et pouvoir gérer de gros volumes de données en stockage et en interrogation. C'est pourquoi nous proposons également une vision distribuée de cette architecture.

## 4.4 Vers une architecture distribuée de PMS

### 4.4.1 Contexte

Avec la maturation des architectures SOA et l'émergence du Cloud computing, les volumes d'échanges des données numériques n'ont cessé d'augmenter. Parallèlement, les sources de provenance se sont multipliés et diversifiés : les applications génèrent leurs traces d'exécution dans des logs applicatifs dédiés ou dans des logs standards des serveurs (serveurs Web, serveurs d'applications ...). Dans un tel écosystème, une vision centralisée des systèmes de gestion de provenance n'est pas adaptée à la distribution et à l'autonomie d'un ensemble d'applications qui génèrent des données de provenance et souhaitent les exploiter.

Effectivement, les applications déployées dans un environnement ouvert et distribué comme le Web sont conçues pour être autonomes pour des raisons évidente de modularité, de sécurité et de confidentialité. Ainsi, l'instanciation de l'architecture logique que nous avons proposé dans une vision purement centralisée ne correspond pas à l'autonomie et la distribution réelle des systèmes impliqués.

Notre vision est une gestion de provenance distribuée où différents fournisseurs de services possèdent chacun leur propre PMS. Pour permettre de décloisonner les PMSs et offrir une vision globale de la provenance inter-fournisseurs, une fédération de PMS permettant leur interrogation conjointe doit être proposée. Cette fédération doit également respecter l'autonomie des PMSs.

Dans ce cas, les interactions entre les systèmes pour le cas Novapost (présenté en centralisé sur la figure 4.3) ne sont plus les mêmes et peuvent être présentés comme illustré sur la Figure 4.4.

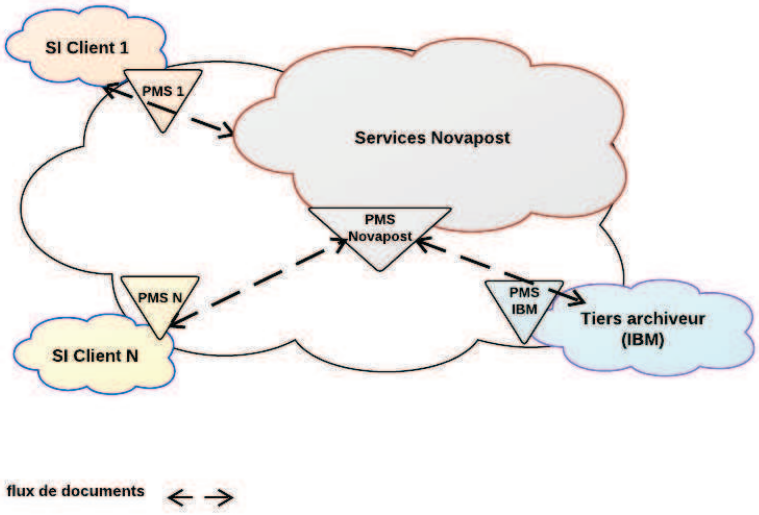


FIGURE 4.4 – Vue sur les interactions entre les systèmes d'une architecture PMS distribuée

### 4.4.2 Choix d'une approche de fédération pour un PMS

Afin de garder l'autonomie des PMSs sources, nous étudions dans cette partie la problématique de traitement distribué des requêtes de provenance. Il s'agit de préserver l'autonomie des différents PMSs et de proposer une approche de distribution permettant de traiter les requêtes de provenance. Nous soulignons que l'autonomie dont nous parlons et que nous souhaitons atteindre est inter-PMSs. Cela veut dire que nous voulons concevoir des systèmes où aucun PMS ne possède une visibilité sur les autres PMSs.

Plusieurs travaux se sont intéressés au traitement distribuées de requêtes où il s'agit d'évaluer une requête sur plusieurs bases de données distribuées [Ozsu et Valduriez, 2011]. L'interrogation distribuée est utilisée principalement pour effectuer, soit de la répartition de charge (load balancing) ou bien de l'intégration des données. Le load balancing est utilisé lorsque nous avons une base de donnée très chargée et qui est souvent sollicitée. Cette technique consiste à dupliquer la base chargée sur plusieurs serveurs. Chaque serveur contient une copie identique à toutes les autres. Dans ce cas de figure, toutes les bases ont le même schéma. Ainsi, les requêtes peuvent être distribuées sur les différents répliques ce qui réduit la charge sur la base principale. Le deuxième scénario est celui de l'intégration des

données où plusieurs bases de données sont distribuées. Dans ce cas de figure, les différentes bases peuvent avoir des schémas différents.

Notre travail se situe dans un contexte d'intégration de données où plusieurs sources de données hétérogènes sont interrogées pour répondre à des requêtes relatives au cycle de vie et à la provenance d'objets numériques, notamment les documents.

Cette problématique d'intégration de données, bien que connue depuis plusieurs années redevient aujourd'hui d'actualité dans un contexte de données liées caractérisé par la multiplication des sources de données, leur hétérogénéité, leur distribution et leurs volumes grandissants. En effet, les techniques d'interrogation des bases de données distribuées actuelles sont fortement liées au monde relationnel et aux requêtes structurées. Ces techniques se basent sur la connaissance de l'emplacement des données à interroger et des modèles correspondants. Freitas et al. [Freitas *et al.*, 2012] jugent que ces techniques ne sont plus adéquates aujourd'hui aux contraintes d'hétérogénéité, de distribution, d'autonomie et de scalabilité. Il suggèrent l'utilisation de méta-données externes pour caractériser les données à interroger ainsi que l'utilisation des approches d'interrogation meilleur effort (Best effort query model) pour les requêtes de matching sémantique.

Gorlitz et al [Gorlitz et Staab, 2011] s'intéressent aussi à l'interrogation des données liées et aux techniques de fédération pour répondre au mieux aux contraintes de distribution et d'autonomie des systèmes qui stockent ce type de données. Ils les assimilent à des bases de données distribuées moins coopératives et qui peuvent être maintenues et mises à jour indépendamment. Ils présentent un ensemble d'exigences pour la conception d'une architecture à base de médiateur pour les systèmes de données liées. Cette architecture se base sur un langage de requêtes déclaratif, un catalogue des modèles, un mapping entre les modèles et les données, un optimiseur de requêtes, un protocole de communication et un mécanisme de classement de résultats.

Quilitz et al. [Quilitz et Leser, 2008] ont traité cette problématique d'interrogation de données liées sur des sources de données hautement distribuées et autonomes. Cependant, ce travail élimine la problématique d'hétérogénéité en ce focalisant uniquement sur les sources de données RDF ayant une interface SPARQL. Un moteur de fédération de requêtes SPARQL appelé DARQ est développé pour assurer un accès transparent à plusieurs sources SPARQL. DARQ implémente un algorithme permettant de réécrire les requêtes SPARQL et d'optimiser leur distribution sur les différentes sources de données.

D'un point de vue architecture, deux grandes familles d'architectures pour la conception des systèmes distribuées sont connues :

– les architectures à base de médiateur (voir Figure 4.6) : dans ces architectures, les systèmes distribués sont intégrés via un composant central appelé médiateur. Ce composant est le seul ayant une vision globale sur tous les systèmes qui composent l'architecture. Il détient un catalogue décrivant tous les schémas ainsi que des méta-données qui décrivent la distribution des données. Selon son niveau de sophistication, le médiateur peut effectuer des traitements sur les requêtes comme l'analyse de

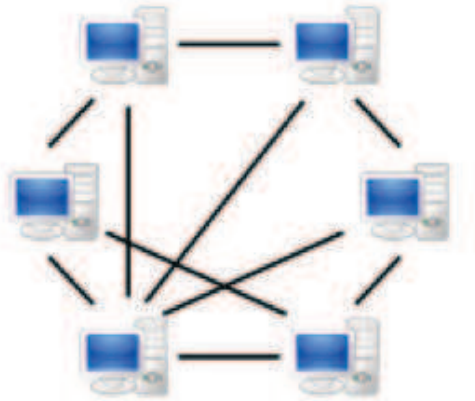

FIGURE 4.5 – Architecture pair à pair

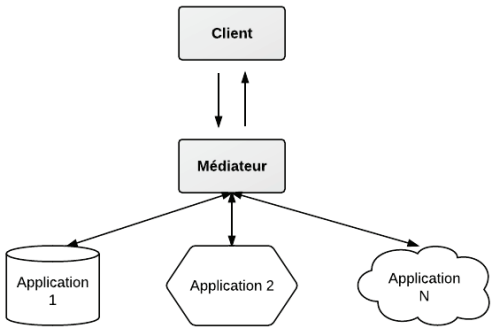


FIGURE 4.6 – Architecture à base de médiateur

schéma, le découpage et la distribution de requêtes, l'agrégation des résultats ...

– les architectures pair à pair (voir Figure 4.5) : dans ces architectures, les différents composants (appelés pairs) sont coopératifs. La gestion des données s'effectue de manière distribuée et collaborative entre tous les pairs. Il n'existe aucun pair central ayant une vision globale sur tous les pairs. Le stockage des données, des indexes et le traitement de requêtes peuvent être distribués sur tous les pairs. Cependant, les pairs nécessitent des mécanismes spécifiques et coûteux pour la découverte des autres pairs, pour le partage et la synchronisation des données.

### 4.4.3 Proposition d'une architecture de PMS fédérée à base de médiateur

Dans une architecture à base de médiateur, ce dernier possède un schéma global sur la distribution des données à travers les différents systèmes. Notre choix d'une architecture à base de médiateur est motivé par le fait que nous devons garder les PMS autonomes, minimiser l'effort d'intégration et ne divulguer qu'un minimum d'information. Le médiateur possède une vision sur tous les PMSs qui composent l'architecture. Par contre, aucun PMS n'a une visibilité sur les autres PMSs. Aussi, chaque PMS ne peut communiquer directement qu'avec le médiateur. Ce dernier fonctionne comme une base de données virtuelle et sert comme une couche d'abstraction. Ainsi, d'un point de vue de l'utilisateur qui effectue des requêtes, la distribution des PMSs derrière le médiateur n'est pas visible et l'interrogation s'effectue comme si un seul PMS existe.

Notre approche, détaillée dans le chapitre 3 introduit un modèle minimal ($MDM$) qui est une brique principale assurant l'interopérabilité entre les modèles de provenance. Elle se base sur le fait que, pour n'importe quel PMS, il est possible de modéliser la provenance selon ce modèle minimal. Ainsi, tous les artifacts de provenance peuvent être décrits en utilisant le $MDM$. Nous proposons d'utiliser cette propriété et de stocker au niveau du médiateur une réplication de toutes les instances $MDM$ qui existent sur tous les PMSs.

Ainsi, nous introduisons une redondance dans le stockage puisque ces instances seront stockés dans deux endroits : sur leur PMS d'origine et sur le médiateur. Bien que coûteuse en terme de stockage, cette approche introduit trois avantages : 1) les requêtes purement MDM (qui portent uniquement sur des propriétés du MDM) peuvent être résolues sur le médiateur sans avoir besoin d'un traitement distribué, 2) pour certaines requêtes, des optimisations sur le schéma et les instances de la requête peuvent être effectuées au niveau du médiateur et permettent de sélectionner au mieux les PMSs sur lesquels la requête sera distribuée. Cette sélection permet de choisir uniquement les PMSs qui peuvent contribuer à la réponse et 3) effectuer la gestion de corrélation entres les artifacts distribués.

Pour assurer ce fonctionnement, ces associations doivent être matérialisées en un ensemble de mappings entre les différents composants, modèles et instances. Ainsi, le médiateur contient les instances, les modèles et les mappings suivants :

- un mapping schéma-source indiquant pour chaque PMS source les schémas ($MDs$) qu'il connait. Ce mapping permet d'éliminer (i.e ne pas interroger) par une analyse statique du schéma de la requête tous les PMSs qui ne correspondent pas au schéma de la requête,
- tous les $MDMs$ instanciés (par réplication depuis les sources),
- un mapping instance-source : c'est le mapping qui établit un lien entre une instance et le PMS source qui l'héberge, Ce mapping sert comme filtre pour une sélection plus fine des PMSs à interroger,
- un mapping schéma-instance : c'est le mapping entre une instance et son schéma ($MD$). Ce mapping sert comme un autre niveau de filtrage des PMSs à interroger puisqu'une instance dont le mapping ne correspond pas au $MD$ ne sera pas considéré dans la réponse,
- les mappings de corrélation, utilisés pour effectuer la corrélation entre les identifiants d'artifacts distribués sur plusieurs PMSs.

En effet, le médiateur est un composant central qui peut être lié à un ou plusieurs PMSs. Chaque PMS héberge un ensemble d'instances. Le lien entre un PMS et ses instances est établi via de mapping instance-source (ce mapping est défini en préfixant chaque identifiant d'artifact par l'identifiant du PMS correspondant comme illustré dans la colonne $instance_{id}$ sur la Table 4.4). Aussi, chaque PMS est lié à un ou plusieurs $MDIs$. Chacun de ces derniers est lié à un modèle de domaine ($MD$) unique et peut contenir plusieurs instances. Ces liens sont établis dans les tables de mapping instance-schéma (voir Table 4.4) et schéma-source (voir Table 4.3).

Ces composants et leurs relations sont présentées sur le diagramme entité-association suivant (voir Figure 4.8). Ces relations qui sont matérialisées dans notre architecture en mappings doivent être gérées par le médiateur et tenues à jour. Le fait de déclarer une nouvelle source de provenance au médiateur implique de :

- déclarer une nouvelle entrée dans la table de mappings schéma-source (avec l'identifiant de la nouvelle source et ses modèles de domaine correspondants),

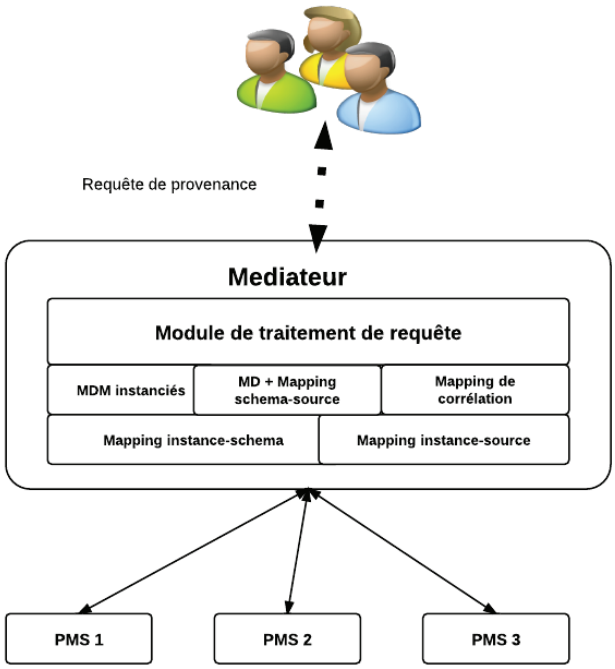


FIGURE 4.7 – Architecture du PMS à base de médiateur

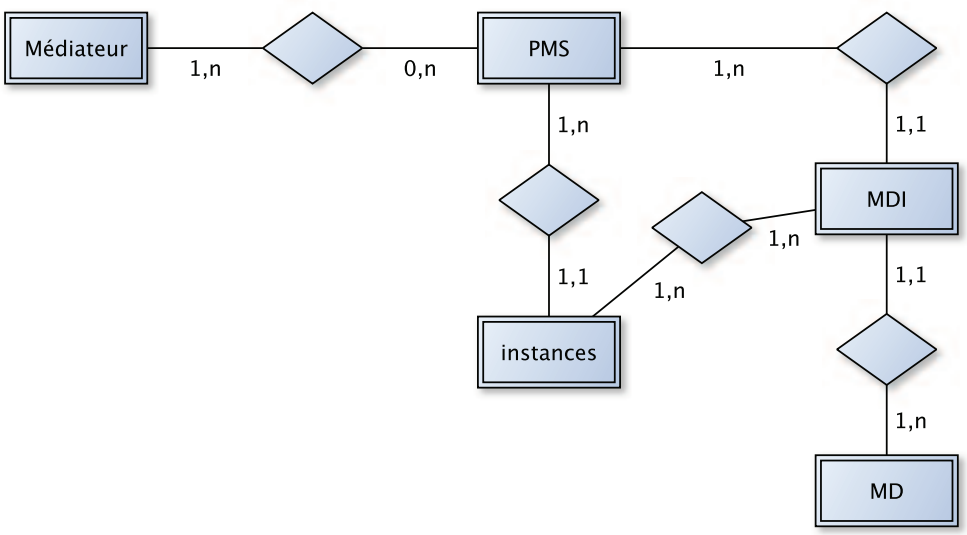


FIGURE 4.8 – Méta modèle pour les composants de l'architecture de PMS fédérée

– ajouter de nouvelles entrées dans la table instance-source (correspondants aux différents identifiants d'artifacts sur cette nouvelle source),
– stocker les instances du $MDM$ de cette source au niveau du médiateur,
– ajouter de nouvelles entrées dans la table instance-schéma (correspondants aux identifiants d'artifacts sur cette source et leurs schémas correspondants),
– ajouter des mappings de corrélation s'il existe des corrélations entre des identifiants

déjà déclarées et les nouveaux identifiants de la nouvelle source.

## 4.5 Évaluation distribuée des requêtes et optimisations

### 4.5.1 Langage de requête logique pour la provenance

Afin d'illustrer et détailler notre approche de distribution, nous devons définir le langage de requêtes utilisé. SPARQL est bien adapté à notre contexte mais est très complexe. Pour cette raison, nous avons choisi de prendre un pseudo-langage basé sur un sous-ensemble de SPARQL.

Nos requêtes de provenance seront formulées avec ce pseudo-langage qui se restreint aux requêtes de sélection (SELECT) uniquement. Ainsi, nous éliminons les formes de requêtes suivantes, supportées dans SPARQL [Prud'hommeaux et Seaborne, 2008] :

1. "CONSTRUCT" : cet opérateur permet de construire un nouveau graphe RDF formé par l'union de tous les triplets qui appartiennent à la séquence solution. Cet opérateur peut être utilisé pour transformer des graphes RDF depuis un modèle vers un autre.
2. "DESCRIBE" : cet opérateur permet de générer une description RDF d'une ressource (voir Exemple de code 4.2).
3. "ASK" : cet opérateur permet de tester si la requête possède une réponse. Il retourne un résultat boolean (voir Exemple de code 4.3).

Ainsi, une requête de provenance dans notre sous-langage est de la forme
**PQ := SELECT ? ( Var+ | * ) DatasetClause WhereClause** avec :

- **SELECT ? ( Var+ | * )** : clause de sélection. Elle permet de construire le résultat de la requête en sélectionnant une ou plusieurs variables utilisées dans le corps de la requête et en appliquant des projections et/ou des agrégations.
- **DatasetClause** (FROM location) : c'est la clause qui spécifie un ou plusieurs points d'entrées vers le (ou les) PMS(s) sur lesquels la requête sera effectuée et qui sont dans notre cas des $MDIs$.
- **WhereClause** : spécifie la (ou les) condition(s) que la requête doit satisfaire. Elle a une structure de la forme : WHERE $condition_i$ op $condition_{i+1}$ ... op $condition_n$ avec op $\in \{ \wedge , \vee \}$ et $condition_i = (expression)$ ( $(property)$ | $(property)$ /^$(property)$ ... $(property))$*|$Filter)$ $(expression)$ avec
  $expression = (var|property|constant)$ et $property$ est une propriété ou un concept d'une ontologie, $var$ est une variable, $constant$ est une constante (valeur numérique, chaine de caractère, ou une expression de chemin ...) et Filter = FILTER $var$ cmp constant et $cmp \in \{ \neq, =, \geq, \leq, >, < \}$ ,

Bien que correctes en SPARQL, les deux requêtes suivantes (voir Exemple de code 4.2 et Exemple de code 4.3) ne sont pas supportées dans le pseudo-langage que nous venons de définir.

```
1 PREFIX nova: <http://track.fr/DM/Novapost/probative.owl>
```

```
2 DESCRIBE ?action
3 WHERE    { ?action nova:performedBy ''nova-archiverAgent''. }
```

Exemple de code 4.2: Requête non supportée dans notre pseudo-langage

```
1 PREFIX nova: <http://track.fr/DM/Novapost/probative.owl>
2 ASK {
3   ?var nova:archivedIn ''IBM-MONTPELLIER''
4 }
```

Exemple de code 4.3: Requête non supportée dans notre pseudo-langage

```
1
2 PREFIX nova: <http://track.Novapost.fr/DM/probative.owl>
3 PREFIX mdm: <http://track.Novapost.fr/DM/mdm.owl>
4 SELECT  ?docId
5 FROM <http://track.fr/idms/Novapost.rdf>
6 WHERE   { ?document mdm:hasDocumentId ?docId .
7           ?document nova:distributionDate ?date .
8               FILTER (
9               ?date > ''2011-12-1''^^xsd:date &&
10              ?date < ''1012-12-31''^^xsd:date )
11                }
```

Exemple de code 4.4: Requête de filtrage supportée dans notre pseudo-langage

```
1 PREFIX mdm: <http://track.Novapost.fr/DM/mdm.owl>
2 SELECT ?linkedAction
3 FROM <http://track.fr/idms/Novapost.rdf>
4 WHERE { ?action mdm:hasOutput ''doc-02-10-C'' .
5           ?action (mdm:hasInput/^mdm:hasOutput)* ?linkedAction.
6           }
```

Exemple de code 4.5: Requête de chemin supportée dans notre pseudo-langage

### 4.5.2 Évaluation distribuée des requêtes

Dans une architecture naïve à base de médiateur, ce dernier n'a aucune capacité de traitement de requêtes. Son rôle consiste simplement à recevoir des requêtes et à les distribuer sur tous les PMSs sans analyser ni leurs schémas, ni leurs contenus. Ensuite, le médiateur récupère les différentes réponses retournées par les PMSs pour construire le résultat final qui sera retourné à l'émetteur de la requête. Ainsi, ce type de médiateur sollicite tous les PMSs pour le traitement de n'importe quel type de requête, ce qui n'est pas une solution optimisée.

Un médiateur plus élaboré va effectuer une analyse du schéma de la requête pour sélectionner uniquement les PMSs pertinents (ceux qui peuvent contribuer à la réponse). Cette analyse de schéma se base uniquement sur le mappings schéma-source pour sélectionner les PMSs qui correspondent au schéma d'interrogation et qui sont susceptibles de retourner une réponse.

Dans notre approche, le médiateur dispose de toutes les instances décrites au niveau du $MDM$. Pour cette raison, nous proposons d'aller plus loin dans la sélection des sources pertinentes en ajoutant **en plus de l'analyse statique du schéma de la requête une analyse de son contenu et une gestion de la corrélation entre les identifiants des artifacts**. Cette analyse se base en plus du mapping schéma-source sur le mapping schéma-instance, le mapping instance-source et aussi sur les informations de corrélation. Cette version essaie de maximiser le travail qui peut être effectué sur le médiateur afin d'éliminer la communication inutile avec les PMSs sources. Elle assure aussi le traitement complet des requêtes $MDM$, l'optimisation du traitement des requêtes par le traitement partiel ou complet des clauses conditions qui portent sur le $MDM$. Elle effectue également la sélection des PMSs, la gestion de corrélation et la réécriture des requêtes ce qui permet une sélection plus pertinente et plus fine des PMSs sur lesquels les requêtes seront réécrites et puis distribuées. Notons que la distribution de provenance peut être selon plusieurs configurations :

- fragmentation horizontale : différents artifact sont distribuées sur plusieurs sources,
- fragmentation verticale : les propriétés d'un même artifact sont distribuées sur plusieurs sources,
- fragmentation hybride (une partie des données est fragmentée horizontalement et une autre partie est fragmentée verticalement)

L'algorithme d'évaluation de requêtes que nous allons présenter dans la section suivante permet de sélectionner un ensemble **minimal** et **complet** de PMSs candidats pour le traitement de requêtes.

#### 4.5.2.1 Algorithme d'évaluation distribuée de requêtes

Pour assurer les fonctionnalités que nous avons décrites, nous proposons un algorithme distribué implémenté sur le médiateur. Son rôle est d'effectuer le traitement et la distribution de requêtes à travers l'ensemble de PMSs. Notre algorithme prend en entrée une requête de provenance pour retourner des identifiants d'artifacts, des attributs ou des chaines de provenance. Notre algorithme présenté ci-dessous (voir Algorithm 1) considère que le traitement de toute requête $PQ$ peut se décomposer en une séquence de trois sous-requêtes dépendantes : $PQ = PQ1 \wedge PQ2 \wedge PQ3$. L'évaluation de $PQ$ se ramène alors à l'évaluation successive de $PQ1$, $PQ2$ et $PQ3$.

L'évaluation de $PQ1$ est un pré-traitement exécuté complètement sur le médiateur. Cette sous-requête utilise uniquement le modèle $MDM$. $PQ2$ est évaluée d'une manière distribuée sur les PMSs sélectionnés et $PQ3$ est est un post-traitement qui produit le résultat final de la requête sur le médiateur. L'ordre de ces requêtes doit être préservé parce qu'en général, $PQ2$ dépend de $PQ1$ et $PQ3$ dépend de $PQ2$. Cependant, selon la requête et la distribution des données, $PQ1$ ou $PQ2$ ou $PQ3$ peut être vide.

Par exemple, pour une requête portant purement sur le $MDM$ (un exemple sera détaillé sur la Figure 4.9), $PQ2$ et $PQ3$ sont vides. Pour les requêtes qui dépendent entièrement

**Algorithm 1:** Évaluation distribuée de requêtes

**Data**: $PQ$, une requête de provenance
**Result**: $RS$, un ensemble d'instances de provenance, d'attributs ou un chemin

```
1  begin
2      < PQ1, PQ2, PQ3 > :=decompose(PQ)
3      r1 :=getCorrelation(∅, PQ1)
4      r2 := execute(r1, PQ1, Mediator)
5      if (PQ2 == ∅) AND (PQ3 == ∅) then
6          return r2
7      r' :=getCorrelation(r2, PQ2)
8      < PMS_ids > := selectSources(PQ2, r2 ∪ r')
9      foreach (pms_id in PMS_ids) do
10         q_i := rewrite(r1, PQ2, pms_id)
11         r_i := execute(∅, q_i, pms_id)
12         r2.add(r_i)
13     r3 := execute(r2, PQ3, Mediator)
14     return r3
```

d'un modèle de domaine spécifique, $PQ1 = \emptyset$. Si $PQ2$ n'est pas vide, alors $PQ3$ ne peut l'être.

Notre algorithme commence par la décomposition de $PQ$. Cette opération est effectuée par l'opérateur ***decompose(PQ)*** qui prend en entrée $PQ$ pour générer les sous-requêtes $PQ1$, $PQ2$ et $PQ3$. Cette décomposition s'effectue en analysant statiquement le schéma de $PQ$. La première sous-requête, $PQ1$ est la partie qui se base sur des attributs du $MDM$ et qui constitue en conjonction avec $PQ2$ la partie condition de $PQ$
(ainsi $condition_{PQ} = condition_{PQ1} \wedge condition_{PQ2}$). Si cette décomposition n'est pas possible, $PQ1$ sera vide.

La deuxième étape du déroulement de l'algorithme est l'exécution de $PQ1$ sur le médiateur. Pour cela, nous introduisons l'opérateur ***execute(R, PQ, $PMS_{id}$)*** qui prend en entrée un ensemble d'identifiants d'artifacts $R$, une requête de provenance et un identifiant de PMS ($PMS_{id}$) qui est l'identifiant du PMS sur lequel $PQ1$ sera exécutée. Ainsi, l'exécution de $PQ1$ sur le médiateur (ligne 4 de notre algorithme) est formulée : execute($\emptyset$, $PQ1$, $Mediator$), $R$ est vide parce que $PQ1$ ne dépend d'aucun résultat en entrée, $PMS_{id}$=$Mediator$ puisque l'exécution de $PQ1$ doit être effectuée uniquement sur le médiateur. L'exécution de cet opérateur génère un ensemble d'attributs du $MDM$ ou un ensemble d'identifiants d'artifacts $MDM$.

Ensuite, l'algorithme effectue une opération de sélection des PMSs sur lesquels $PQ2$ sera distribuée. Cette sélection est faite par l'opérateur ***selectSources(PQ,R)*** (ligne 8 de l'algorithme) qui prend en entrée une requête de provenance (dans ce cas, il s'agit de la sous-requête $PQ2$ en entrée) et un ensemble d'identifiants d'artifacts (ce sont les identifiants d'artifacts produits par l'exécution de $PQ1$). Le rôle de cet opérateur est

de sélectionner l'ensemble ***minimal*** et ***complet*** des identifiants de PMSs où $PQ2$ sera exécutée. Cette sélection est basée sur les mappings schéma-source, instance-source ainsi que sur les informations de corrélation. Cet opérateur de sélection doit retourner tous les identifiants de PMSs pertinents relativement à la requête (critère de complétude) et seulement ceux là (critère de minimalité).

Ensuite, pour chaque PMS sélectionné, une réécriture de $PQ2$ est effectuée. Elle est spécifique à chaque $PMS$ et dépend des schémas et des instances que chaque nœud contient. C'est l'opérateur ***rewrite(***$R$,$PQ$,$PMS_{id}$***)*** qui l'effectue (ligne 10 de l'algorithme). Cet opérateur de réécriture prend en entrée des identifiants d'artifacts (résultats d'exécution d'une autre sous-requête), une sous-requête et un identifiant de PMS ($PMS_{id}$) sur lequel la nouvelle sous-requête $PQ2'$ doit être exécutée. Dans la même boucle et pour chaque PMS sélectionné, l'algorithme exécute la requête $PQ2'$ sur le $PMS$ en question (ligne 11 de l'algorithme).

Finalement, les différents $PMSs$ retournent les résultats d'exécution au médiateur. A minima, $PQ3$ réalise l'union et/ou la jointure des résultats retournés par les différents $PMSs$ selon la nature de $PQ$. En fonction de $PQ$, le médiateur peut également effectuer les projections et les agrégations nécessaires (ligne 13 de l'algorithme).

#### 4.5.2.2 Présentation du jeu de données de test

Pour illustrer le fonctionnement de notre algorithme sur divers exemples, nous proposons un jeu de données de test (fourni par Novapost). Ce jeu de données est issu de trois sources ou PMS (Customer1, Nova1 et Nova2). Nova1 génère des instances selon trois modèles de domaine : $MDM$, $MD_{Novapost}$ et $MD_{Customer1}$ (voir Table 4.3).

L'instance cust1-doc753 appartient à la source Nova2 et est décrite par le modèle $MD_{Customer1}$. Il s'agit du modèle le plus spécialisé pour cette instance sur cette source (voir Table 4.4). Cette même instance est équivalente (*sameAs*) à l'instance nova-cust1-12-05-011 gérée par la source Nova1 et décrite par le modèle $MD_{Novapost}$ (voir Table 4.2).

#### 4.5.2.3 Exemple 1 : traitement d'une requête MDM

L'utilisateur s'interroge sur toutes les actions qui ont utilisé le document "cust5-p082010" comme entrée. Cela s'exprime sous forme de la requête PQ : SELECT ?action FROM Nova1, Nova2, Customer1 WHERE ?action mdm :hasInput "cust5-p082010" . Son traitement par notre algorithme est illustré sur la Figure 4.9 qui suit. Il se déroule selon les étapes suivantes :

1. $decompose(PQ)$ : décompose la requête $PQ$ en trois sous-requêtes $PQ1$, $PQ2$ et $PQ3$. $PQ1$ est la partie $MDM$ de $PQ$. Pour cet exemple, $PQ1$=$PQ$, $PQ2$= $\emptyset$ et $PQ3$= $\emptyset$, avec $PQ1$= SELECT ?action WHERE { ?action mdm :hasInput "cust5-p082010" }
2. $execute(\emptyset, PQ1, Mediator)$ : exécution de $PQ1$ sur le médiateur pour générer $RS1$ = <act-dist566,act-arc895,act-arc895>
3. retourner le résultat final de la requête : $RS = RS1$.

| $initial_{id}$ | $corresponding_{ids}$ |
|---|---|
| Nova1 :nova-cust1-12-05-011 | Nova2 :cust1-doc753 |
| Customer1 :cust3-092010 | Nova1 :cust1-doc-3-092010 |
| Customer1 :cust3-092010-s1 | Nova1 :nova-sss-140 |
| Customer1 :cust3-092010-s2 | Nova1 :nova-sss-141 |
| Customer1 :cust3-092010-s12 | Nova1 :nova-sss-142 |

TABLE 4.2: Mapping de corrélation

| $PMS_{id}$ | $schéma_{id}$ |
|---|---|
| Customer1 | $MDM$, $MD_{Customer1}$ |
| Nova1 | $MDM$, $MD_{Novapost}$, $MD_{Customer1}$ |
| Nova2 | $MDM$, $MD_{Novapost}$ |

TABLE 4.3: Mapping schéma-source

| $instance_{id}$ | $schéma_{id}$ |
|---|---|
| Customer1:doc-02889 | $MD_{Customer1}$ |
| Nova1:nova-c22-02889-463 | $MD_{Novapost}$ |
| Nova2nova-c22-02889-477 | $MD_{Novapost}$ |
| Nova2nova-c22-02889-491 | $MD_{Novapost}$ |
| Nova1:nova-cust1-12-05-011 | $MD_{Novapost}$ |
| Nova2cust1-doc753 | $MD_{Customer1}$ |
| Nova1:nova-sss-140 | $MD_{Novapost}$ |
| Nova1:nova-sss-141 | $MD_{Novapost}$ |
| Nova1:nova-sss-142 | $MD_{Novapost}$ |
| Nova2cust2-911-37 | $MD_{Novapost}$ |
| Nova2cust2-911-39 | $MD_{Novapost}$ |
| Nova2cust2-911-49 | $MD_{Novapost}$ |
| Customer1:cust3-092010-s1 | $MD_{Customer1}$ |
| Customer1:cust3-092010-s2 | $MD_{Customer1}$ |
| Customer1:cust3-092010-s12 | $MD_{Customer1}$ |
| Customer1:cust3-092010 | $MD_{Customer1}$ |
| Nova2archived13-C | $MD_{Novapost}$ |
| Nova2archived97-C | $MD_{Novapost}$ |
| Nova1:cust1-doc-3-092010 | $MD_{Novapost}$ |
| Nova1:cust5-p082010 | $MDM$ |
| Nova1:act-dist566 | $MDM$ |
| Nova1:act-arc895 | $MDM$ |
| Nova1:act-arc895 | $MDM$ |

TABLE 4.4: Mapping instance-schéma (encapsulant le mapping instance-source)

On voit bien ici que le traitement de cette requête peut se faire complètement sur le médiateur.

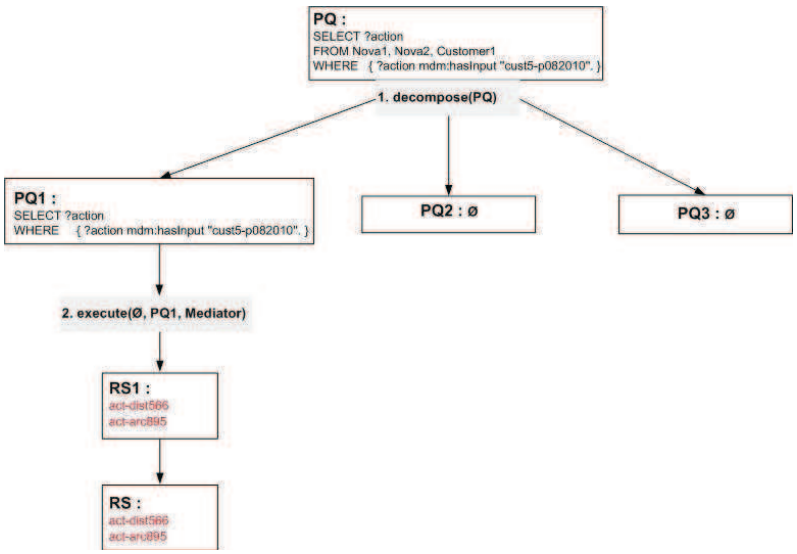


FIGURE 4.9 – Déroulement du traitement d'une requête $MDM$

#### 4.5.2.4 Exemple 2 : traitement d'une requête dont la condition peut être résolue entièrement sur le médiateur

Sélectionner les empreintes et le statut d'archivage des documents issus d'opérations sur le document identifié par "doc-02889" et qui ont été effectuées par le service de distribution électronique "Novapost". Ceci s'exprime par la requête suivante : SELECT ?sha256 ?status From Nova1, Nova2, Customer1 WHERE {
?action mdm :hasInput "doc-02889" .
?action mdm :hasOutput ?documents .
?documents nova :hasSHA256 ?sha256 .
?documents nova :hasStaus ?status . }
La clause condition de cette requête porte uniquement sur des attributs du $MDM$ et peut être donc résolue entièrement sur le médiateur. Cependant, la clause de sélection porte sur des propriétés spécifiques à $MD_{Novapost}$ et donc la requête (une réécriture de la requête originale) doit être distribuée sur les $PMSs$.

Le traitement de cette requête est illustré sur la Figure 4.10. Son déroulement s'effectue selon les étapes suivantes :

1. decompose($PQ$) : décompose la requête $PQ$ en trois sous-requêtes $PQ1$, $PQ2$ et $PQ3$. $PQ1$ est la partie $MDM$ de la requête, (voir Figure 4.10 -PQ1). $PQ2$ est la partie qui dépend du $MD_{Novapost}$ (présenté dans le chapitre 3) et forme en conjonction avec $PQ1$ la partie condition de $PQ$ (voir Figure 4.10 -PQ2). $PQ3$ effectue une opération

d'union des résultats d'exécution des sous-requêtes issues de $PQ2$ (réécritures de $PQ2$ sur les $PMSs$ sélectionnés).

2. execute($\emptyset, PQ1, Mediator$) : cette exécution ne dépend pas d'un ensemble d'artifacts en entrée ($R = \emptyset$). Elle est effectuée sur le médiateur et génère RS1= <nova-c22-02889- 463, nova-c22-02889- 477, nova-c22-02889- 491> . Cette étape met en évidence le rôle du médiateur dans le traitement de la condition de la requête. Ce traitement est complet vu que le traitement de la condition s'effectue complètement sur le médiateur et se base sur les instances du $MDM$ stockées sur le médiateur.

3. injecter RS1 comme une entrée de PQ2 pour sélectionner les $PMSs$ candidats. Cette sélection se base sur les mappings schéma-source, instance-source et schéma-instance pour les instances de RS1. Elle permet d'éliminer la source Customer1 (qui ne contient pas d'instances correspondantes au modèle $MD_{Novapost}$) et de ne garder que les sources Nova1 et Nova2. Dans cet exemple, si on n'utilise pas le mapping de schéma (schéma-source), on doit garder également la source Customer1.

4. rewrite($RS1$, $PQ2$, $Nova1$) : pour la première source sélectionnée et en se basant sur les mappings instance-source, générer $PQ2$-1 = SELECT ?sha256 ?status FROM Nova1 WHERE ( ?documents nova :hasSHA256 ?sha256)
AND ( ?documents nova :hasStatus ?status) BINDINGS { ?documents (nova-c22-02889- 463)}

5. rewrite ($RS1$, $PQ2$, $Nova2$) : pour la deuxième source sélectionnée, générer $PQ2$-2 = SELECT ?sha256 ?status FROM Nova1 WHERE ( ?documents nova :hasSHA256 ?sha256) AND ( ?documents nova :hasStatus ?status) BINDINGS { ?documents (nova-c22-02889- 477) (nova-c22-02889- 491)} .
Notons que la réécriture est également plus précise puisque nous injectons des informations provenant des mappings d'instances (instance-source). La réécriture permet aussi de spécifier pour chaque source sélectionnée l'ensemble des identifiants pertinents (comme "nova-c22-02889- 463" sur la source Nova1 par exemple).

6. execute($\emptyset$,$PQ2$-1,$Nova1$) : exécuter $PQ2$-1 sur la première source sélectionnée (Nova1) pour générer le résultat $RS2$-1,

7. execute($\emptyset$, $PQ2$-2, Nova2) : exécuter $PQ2$-2 sur la deuxième source sélectionnée (Nova2) pour générer le résultat $RS2$-2,

8. injecter les deux résultats d'exécution obtenus ($RS2$-1 et $RS2$-2) comme une entrée pour $PQ3$,

9. execute (<$RS2$-1, $RS2$-2>, $PQ3$, Mediator) : exécuter $PQ3$ sur le médiateur en se basant sur les résultats injectés. Cette opération permet de constituer le résultat final de la requête en effectuant l'union de $RS2$-1 et $RS2$-2.

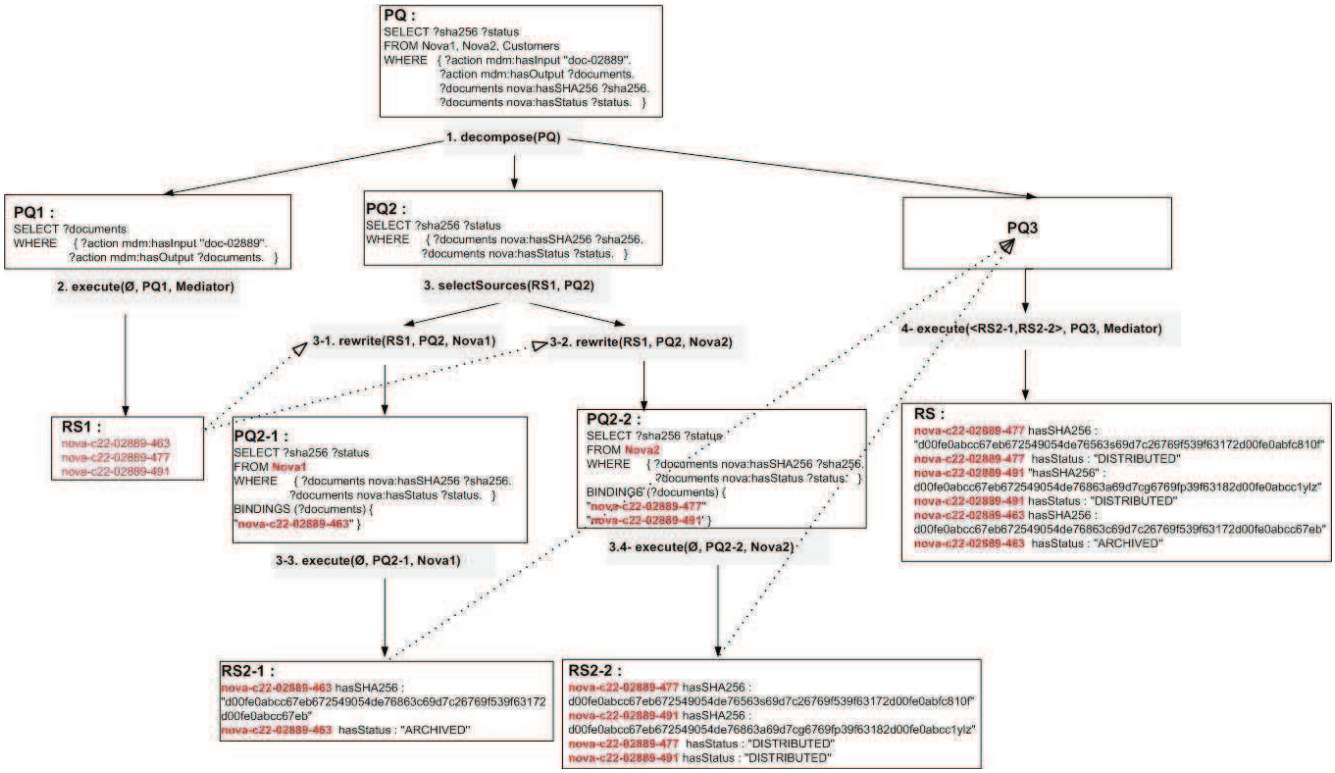


FIGURE 4.10 – Exemple de requête où le traitement de la condition s'effectue totalement sur le médiateur

#### 4.5.2.5 Exemple 3 : traitement d'une requête dont la condition peut être résolue partiellement sur le médiateur

Sélectionner les empreintes et le statut d'archivage du (ou des) document(s) qui ont permis de générer le document "cust2-092011" suite à une opération de fusion effectuée par un (ou des) acteur(s) authentifié(s) en les groupant par identifiant de document.

En SPARQL, cette requête s'exprime de la manière suivante :

SELECT ?sha256 ?status ?documentId FROM Nova1, Nova2, Customer1 WHERE {

?action mdm :hasOutput "cust2-092011" .

?action nova :hasType "COMPRESSION-7Z" .

?action nova :hasStatus ?status .

?documents nova :hasSHA256 ?sha256 . }

Cette requête possède une clause condition qui peut être résolue partiellement sur le médiateur puisqu'elle porte sur des propriétés du $MDM$ et des propriétés spécifiques au domaine Novapost (l'opération de distribution électronique et les acteurs authentifiés). Quant à la clause de sélection, elle porte sur des propriétés spécifiques à ce même modèle de domaine ($MD_{Novapost}$) et doit donc être distribuée sur les PMSs.

Le traitement de cette requête est illustré sur la Figure 4.11. Son déroulement s'effectue

selon les étapes suivantes :

1. decompose($PQ$) : décompose la requête $PQ$ en trois sous-requêtes $PQ1$, $PQ2$ et $PQ3$. $PQ1$ est la partie qui dépend des attributs du $MDM$ (voir Figure 4.11 -PQ1) : $PQ2$ est la partie qui dépend de $MD_{Novapost}$ (voir Figure4.11 -PQ2) :
   $PQ2$ forme en conjonction avec $PQ1$ la clause condition de $PQ$. $PQ3$ est l'agrégation des résultats de $PQ2$ : SELECT ?sha256 ?status GROUP BY ?documentId (voir Figure 4.11 -PQ3),
2. execute($\emptyset, PQ1, Mediator$) : cette exécution ne dépend pas d'un ensemble d'artifacts en entrée ($R = \emptyset$). Elle est effectuée sur le médiateur et génère RS1= <cust2-911-37, cust2-911-39, cust2-911-49>,
3. selectSources($RS1, PQ2$) : injecter $RS1$ et $PQ2$ pour sélectionner les sources pertinentes. Cette sélection se base sur les mappings schéma-source (ce qui permet d'éliminer la source Customer1 qui ne contient pas des instances correspondantes au schéma de la requête, à savoir $MD_{Novapost}$ ), schéma-instance et instance-source. Ainsi, la source Nova1 est supprimée (par mapping instance-source). La seule source sélectionnée dans ce cas est Nova2 car elle correspond à un mapping instance-source (voir colonne $instance_{id}$ sur la Table 4.4 ) et au mapping schéma-instance (voir Table 4.4).
4. rewrite($RS1$, $PQ2$, $Nova2$) : pour la source sélectionnée et en se basant sur les mapping instance-source, réécrire la requête pour générer $PQ2$-1 (voir Figure 4.11 PQ2-1) Cette réécriture assure que la source sélectionnée (Nova2) correspond au schéma de la requête et qu'elle contient des instances qui correspondent à la clause condition.
5. execute($\emptyset$, $PQ2$-1, $Nova2$) : exécuter $PQ2$-1 sur le PMS sélectionné (Nova2) pour générer le résultat intermédiaire $RS2$-1,
6. execute ($RS2$-1, $PQ3$, $Mediator$) injecter le résultat d'exécution intermédiaire ($RS2$-1) comme une entrée pour $PQ3$ et exécuter ensuite $PQ3$ sur le médiateur pour effectuer l'agrégation et générer le résultat final de la requête $RS$.

#### 4.5.2.6 Exemple 4 : traitement d'une requête effectué complètement sur les sources

Sélectionner l'emplacement et le statut d'archivage d'un document qui a été envoyé par une entreprise cliente et dont on connait uniquement son empreinte SHA256.

Cette requête possède une clause condition qui ne peut pas être résolue ni partiellement, ni complètement sur le médiateur parce que la sélection et la condition portent sur des propriétés spécifiques au modèle $MD_{Novapost}$. Elle est formalisée en SPARQL comme suit :

```
SELECT ?location ?status WHERE {
?document nova :archivedIn ?location .
?document nova :hasStatus ?status .
```

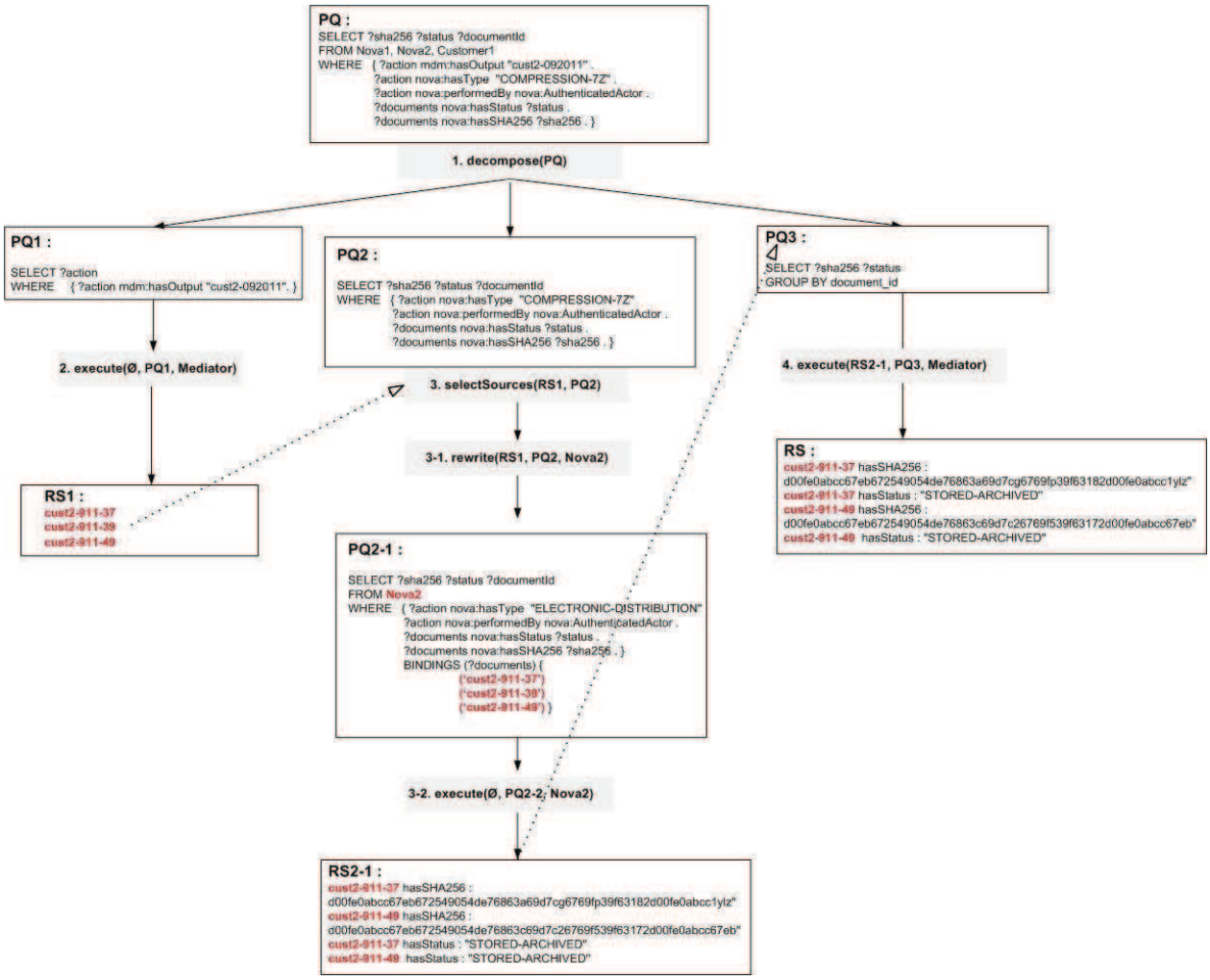


FIGURE 4.11 – Requête avec traitement partiel de la condition sur le médiateur

?document nova :hasSHA256 '8016453C6A0CCB04E84E03F9D7615D24D247F4CD4F764268FB1B1C717E1CA'
. }

Le déroulement du traitement de cette requête est illustré sur la figure 4.12. Le seul filtrage que le médiateur peut effectuer dans ce cas est un filtrage selon l'emplacement des modèles, basé sur le mapping schéma-source. Il permet d'éliminer la source Customer1. Le traitement de la requête s'effectue alors selon les étapes suivantes :

1. decompose($PQ$) : décompose la requête $PQ$ en trois sous-requêtes $PQ1$, $PQ2$ et $PQ3$. Dans ce cas, $PQ1 = \emptyset$, $PQ2$=$PQ$ et $PQ3$ effectue la fusion des résultats intermédiaires des réécritures de $PQ2$.
2. selectSources($\emptyset$, $PQ2$) : en se basant sur les mapping schéma-source, sélectionner les sources qui correspondent au modèle $MD_{Novapost}$ : Nova1 et Nova2.
3. rewrite($\emptyset$, $PQ2$, $Nova1$) : réécrire la requête pour la première source sélectionnée

Nova1 (voir Figure 4.12 PQ2-1),

4. rewrite(∅, $PQ2$, $Nova2$) : réécrire la requête pour la deuxième source sélectionnée Nova2 (voir Figure 4.12 PQ2-2)„ Pour cette requête, le réécriture spécifie les sources en se basant sur le mapping schéma-source sans pouvoir raffiner davantage la sélection des sources. Ceci est du au fait que la clause condition ne spécifie pas des identifiants d'artifacts permettant de guider notre processus de sélection des sources.
5. execute(∅, $PQ2$-1, $Nova1$) : exécuter $PQ2$-1 sur la source Nova1 pour générer le résultat intermédiaire $RS2$-1,
6. execute(∅, $PQ2$-2, $Nova2$) : exécuter $PQ2$-2 sur la source Nova2 pour générer le résultat intermédiaire $RS2$-2,
7. execute(<$RS2$-1,$RS2$-2>,$PQ3$, $Mediator$) permet d'établir le résultat final formé par l'union des résultats intermédiaires $RS2$-1 et $RS2$-2 (voir Figure 4.12- RS).

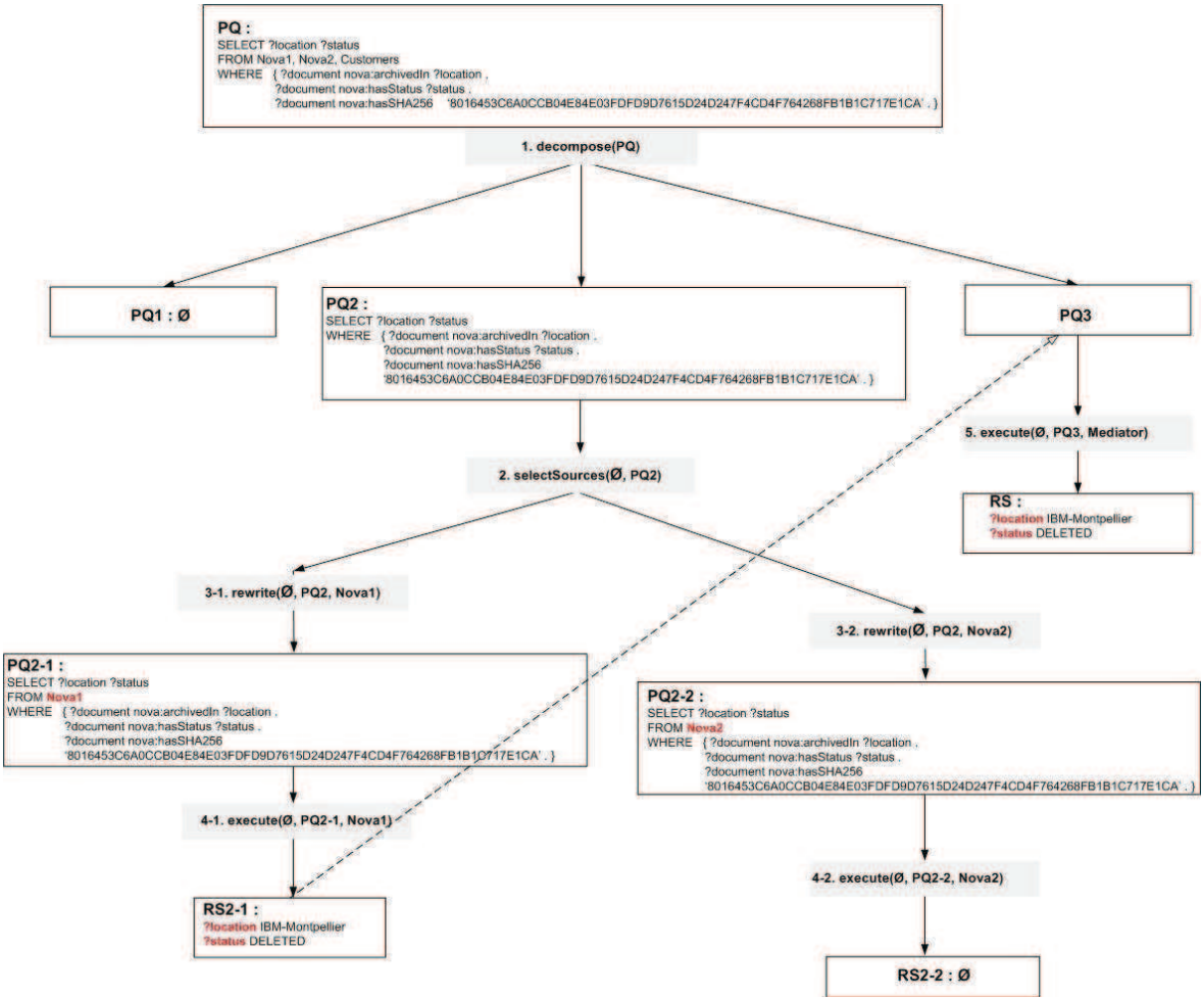


FIGURE 4.12 – Requête avec traitement complet sur les sources

#### 4.5.2.7 Gestion de corrélation dans le traitement de la requête

Dans notre contexte, la provenance d'un même artifact peut être distribuée sur plusieurs PMSs. L'identifiant d'un même artifact peut être (et est souvent) différent d'un PMS à l'autre puisque les systèmes d'identification sont différents. Pour cette raison, la corrélation d'identifiants est nécessaire pour obtenir des réponses pertinentes et complètes. Cette problématique de corrélation est un grand défi pour les systèmes de gestion de provenance dans un environnement distribué. Afin d'établir des liens entre les artifacts distribués sur différentes sources de provenance, nous avons introduit dans notre approche la notion de mapping de corrélation. Dans notre architecture distribuée à base de médiateur, nous avons proposé de stocker cette information au niveau du médiateur. Elle indique que l'identifiant d'un artifact sur un PMS est identique à un autre identifiant sur un autre PMS et doit être fournie par un administrateur. Elle sert pour établir des liens directs de corrélation entre artifacts permettant d'inférer de nouvelles propriétés et relations. Cette gestion de corrélation entre artifacts est effectuée avec l'opérateur getCorrelation($RS, PQ$) qui identifie des artifacts pertinents pour une requête donnée en effectuant de la corrélation d'identifiants. Cette inférence sur les modèles peut se ramener parfois à des réécritures de requêtes qui prennent en compte les relations déduites entre les modèles à partir de leurs instances.

Ce mode de traitement avec gestion de corrélation n'est pas le mode de fonctionnement par défaut de notre algorithme (voir Algorithm 1 - ligne 3 et ligne 6). Pour l'utiliser, il faut soumettre la requête en mode "corrélation".

**Exemple de traitement d'une requête avec gestion de corrélation** : Par exemple, un auditeur effectue des vérifications sur le déroulement d'une opération qui génère des documents à valeur probante. Cette vérification se base sur le fait que les documents issus de cette opération ont été archivés dans un système sécurisé et possèdent des méta-données de preuve qui leurs sont rattachés. Ainsi, une requête effectuant cette vérification se base sur les attributs à valeur probante pour les documents issus de cette opération. Pour illustrer l'intérêt de la gestion de corrélation, nous prenons l'exemple du document "cust3-092010". Pour ce document, la requête SPARQL correspondante à cette vérification est formalisée comme suit :

```
SELECT ?xmlTrack ?documentId FROM Nova1, Nova2, Customer1 WHERE {
?action mdm :hasInput "cust3-092010" .
?documents nova :archivedIn "IBM-DATACENTER".
?documents nova :hasXMLTrack ?xmlTrack .}
```

Avec la distribution de nos données de test (voir Table 4.4), notre requête qui s'intéresse au document "cust3-092010" et qui porte sur les propriétés du modèle $MD_{Novapost}$ n'a pas de réponse. Sans gestion de corrélation, la réponse retournée sera $RS = \emptyset$ parce que l'artifact

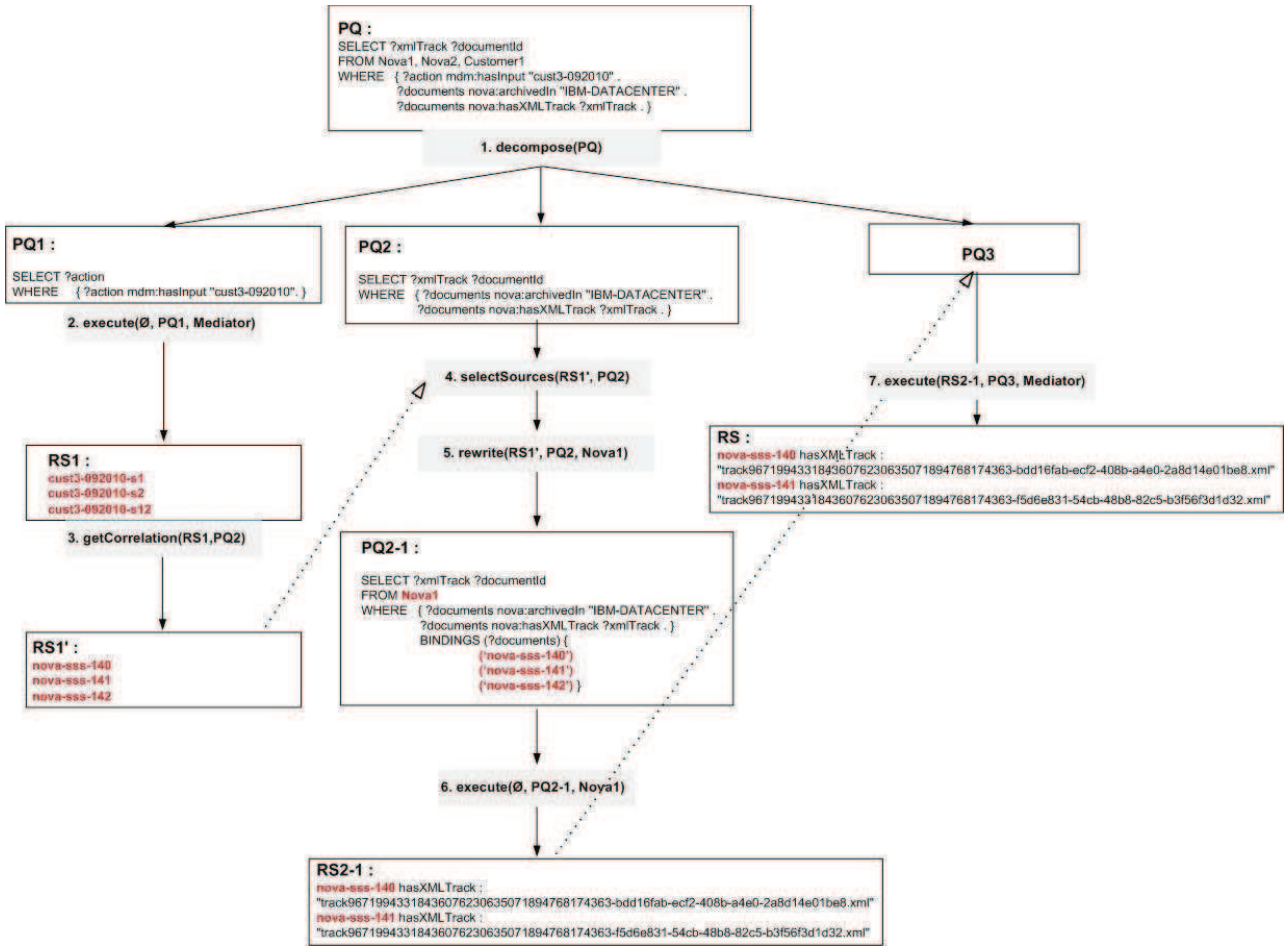


FIGURE 4.13 – Traitement d'une requête avec gestion de corrélation

identifié par "cust3-092010" génère des artifacts qui n'ont pas un $MDI$ correspondant au modèle ($MD_{Novapost}$). Cependant, cette réponse n'est pas vraiment exacte. En effet, l'artifact identifié par "cust3-092010" génère des documents <cust3-092010-s1, cust3-092010-s2, cust3-092010-s12> Cette information est obtenue aprés l'éxecution de $PQ1$ (voir Figure 4.13 - PQ1) Ces documents qui sont corrélés aux documents <nova-sss-140, nova-sss-141, nova-sss-142> (voir 4.2) et qui correspondent au $MDI_{Novapost}$ (voir Table 4.4). Ainsi, la réécriture de la requête avec les nouveaux identifiants et son exécution sur le (ou les) PMS(s) correspondant(s) permet de retourner une réponse pertinente qui correspond au cycle de vie réel du document. Le traitement de cette requête par notre algorithme s'effectue selon les étapes suivantes :

1. decompose($PQ$) : décompose la requête $PQ$ en trois sous-requêtes $PQ1$, $PQ2$ et $PQ3$. $PQ1$ est la partie $MDM$ de la requête, (voir Figure 4.9 -PQ1) : SELECT ?document WHERE ?action mdm :hasInput "cust3-092010"

   $PQ2$ est la partie qui dépend du modèle $MD_{Novapost}$ (voir Figure 4.9 -PQ2) et qui forme en conjonction avec $PQ1$ la partie condition de $PQ$. $PQ3$ assure la suppression

des doublons liés à la corrélation. En effet, à cause des corrélations, on peut retourner à l'utilisateur un (ou plusieurs) artifact(s) qui sont corrélés et donc identiques.

2. execute($\emptyset, PQ1, Mediator$) : la première étape est l'exécution de $PQ1$. Évidemment, cette exécution ne dépend pas d'un ensemble d'artifacts en entrées ($R = \emptyset$). Elle est effectuée sur le médiateur. Elle génère $RS1$ : documents_ids= <cust3-092010-s1, cust3-092010-s2, cust3-092010-s12>.

3. getCorrelation($RS1, PQ2$) : Les artifacts obtenus ($RS1$) ne possèdent pas un mapping sur le modèle de domaine (schéma) de la requête. Cette opération permet d'obtenir les identifiants d'artifacts équivalents aux identifiants d'artifacts générés et correspondants au modèle d'interrogation de $PQ$. Son résultat est $RS1'$= <nova-sss-140, nova-sss-141,nova-sss-142> et se base sur le mapping de corrélation (voir Table 4.2).

4. selectSources($PQ2, RS1'$) : sélection des sources pertinentes en se basant sur les nouvelles instances obtenues par corrélation ($RS1'$). C'est la source Nova1 qui sera sélectionnée en se basant sur le mapping instance-source et le mapping schéma-instance (voir Table 4.4 )

5. rewrite($RS1$, $PQ2$, $Nova1$) : pour la source Nova1 et en se basant sur les mappings instance-source, générer $PQ2\text{-}1$ = SELECT ?XMLTrack ?status FROM Nova1 WHERE {
   ?document nova :hasXMLTrack ?track .
   ?documents nova :hasStatus ?status .
   BINDINGS ?documents
   {(nova-sss-140)
   (nova-sss-141)
   (nova-sss-142)})
   }

6. execute($\emptyset$,$PQ2\text{-}1$,$Nova1$) : exécuter $PQ2\text{-}1$ sur la seule source sélectionnée (Nova1) pour générer le résultat $RS2\text{-}1$,

7. execute($RS2\text{-}1$, $PQ3$, $Mediator$) : exécution de $PQ3$ sur le médiateur en se basant sur le résultat intermédiaire $RS2\text{-}1$. Pour cette exécution, il n'y a ni union, ni jointure, ni calcul d'agrégats. $PQ3$ vérifie s'il y a des doublons (vu qu'il y a eu corrélation entre plusieurs instances), les supprime s'ils existent et retourne le résultat final $RS$.

### 4.5.3 Synthèse

Dans cette section, nous avons présenté une architecture distribuée de PMS basée sur un médiateur. Cette architecture à base de médiateur nous a permis de préserver l'autonomie inter-PMSs et d'assurer une vision globale de la provenance inter-fournisseurs.

Le traitement des requêtes s'effectue par un algorithme de distribution implémenté au niveau du médiateur. Notre algorithme permet de faire en plus de l'analyse statique du

schéma de la requête une analyse basée sur les instances ainsi qu'une gestion de corrélation entre les artifacts. Notre algorithme suppose la duplication des instances du $MDM$ sur le médiateur ce qui lui permet de traiter les requêtes qui portent uniquement sur des attributs du $MDM$, de traiter la clause condition (partiellement ou complètement) pour certains types de requêtes et de mieux optimiser leur distribution sur les PMSs sources pour celles qui dépendent d'un $MD$ donné.

## 4.6 Conclusion

Dans ce chapitre, nous avons contribué au troisième objectif de notre thèse en proposant deux architectures de PMS : la première est une architecture logique et générique et la deuxième peut être vue comme une instantiation dans un environnement distribué de cette architecture logique.

L'architecture logique que nous avons proposé dans la première partie de ce chapitre est indépendante des choix technologiques d'implémentation et de déploiement. Elle présente la mise en œuvre des concepts et des fonctionnalités introduits dans notre approche (présentés dans le chapitre 3) et couvre les étapes de collecte, de gestion et d'interrogation de la provenance. Elle se compose principalement d'un module de collecte et de gestion réservé aux administrateurs, d'un module de stockage et d'un module d'interrogation.

Dans la deuxième partie de ce chapitre, nous avons présenté une architecture distribuée d'un PMS. Lors de la conception de cette architecture, l'aspect autonomie inter-PMSs était notre première préoccupation. En effet, cette contrainte correspond à l'autonomie et la distribution réelle des sources de provenance. Notre architecture se base sur un médiateur ayant une vision globale sur tous les PMSs ainsi que des capacités de traitement de requêtes. Nous avons surtout mis l'accent sur l'indépendance des PMSs : c'est uniquement le médiateur qui possède une vision globale et qui assure le lien entre les différents PMSs. Certes, notre architecture ne garantit pas une autonomie totale des PMSs (puisque ces derniers divulguent leurs modèles, un certain nombre d'instances et de mappings au médiateur) mais elle assure l'autonomie inter-PMSs. Notre architecture assure aussi le traitement distribuée des requêtes. Cette capacité de traitement et de distribution est assurée par un algorithme implémenté au niveau du médiateur. Il se base sur un catalogue des schémas global (les $MDs$), sur une duplication des instances du $MDM$ et sur un ensemble de mappings entre les schémas, les instances et les sources ainsi que sur un ensemble d'informations de corrélation. Le travail proposé dans ce chapitre a été validé par une publication dans une conférence internationale [Sakka et Defude, 2012a].

Dans le chapitre 5, nous évaluerons notre approche de modélisation de provenance en proposant une validation fonctionnelle des modèles proposés dans notre approche et une validation non-fonctionnelle effectuée sur les données de Novapost et qui implémente deux prototypes de PMSs basés sur deux technologies de stockage différentes. Nous présentons des tests de performance et de scalabilité sur les logs de Novapost contenant des informations

de provenance sur ses processus de gestion, d'archivage et de gestion documentaire.

*Chapitre 5*

# Validation, implémentation et expérimentations

*"D'impossibles à imaginaires, d'imaginaires à complexes. Combien d'idées, de systèmes politiques, de théories, de procédés ont suivi ce chemin pour devenir réalité !"*

Denis Guedj, "Le théorème du perroquet"

## Sommaire

## 5.1 Introduction

Dans le chapitre 4, nous avons présenté une architecture logique de PMS et un algorithme de distribution pour le traitement de requêtes de provenance dans un contexte distribué où plusieurs sources de provenance sont impliquées. Dans ce chapitre, nous nous intéressons à la validation de nos différentes propositions présentées dans les chapitres précédents. Nous nous focalisons sur la validation fonctionnelle de notre approche de modélisation, la présentation du contexte industriel et des développements réalisés chez Novapost et sur la validation non-fonctionnelle de la scalabilité de notre architecture de PMS.

Ce chapitre est structuré comme suit. Tout d'abord, nous présentons dans la section 5.2 des éléments de validation de la généricité de notre approche de modélisation présentée dans le chapitre 3. Nous présentons dans la section 5.3 le cadre industriel dans lequel une grande partie de notre travail a été effectuée. Nous décrivons la solution de dématérialisation de Novapost et ses différents modules et nous présentons les Web services REST que nous avons développé pour générer des données de provenance sur les processus de gestion documentaire chez Novapost. Ensuite, nous étendons cette partie avec une description de notre technique d'intégration de la provenance générée via des services dédiés dans notre framework. Dans la section 5.4, nous décrivons le travail que nous avons effectué pour tester les technologies les mieux adaptées pour le module de stockage de provenance. Nous réalisons des tests sur deux technologies de bases de données différents (RDF Store et NoSQL) et nous discutons pour chacune d'entre elles la représentation de la provenance, son interrogation, l'expressivité des requêtes ainsi que la scalabilité du PMS. Nous présentons à la fin de cette section une synthèse sur nos différentes expérimentation. Enfin, la section 5.5 conclut ce chapitre.

## 5.2 Validation fonctionnelle

Dans le chapitre 3, nous avons proposé un modèle de domaine riche, spécialisant notre modèle minimal $MDM$ pour la modélisation de la provenance dans le domaine de l'archivage légal et correspondant aux besoins de Novapost. Pour valider l'aspect générique de notre modèle, nous allons l'appliquer sur un autre contexte, celui de la facturation d'applications/services dans un environnement de Cloud computing. Aujourd'hui, le nombre de fournisseurs de services dans le Cloud ne cesse d'augmenter. Ces fournisseurs peuvent être des personnes, des organisations ou des entités qui proposent un ou plusieurs services à des consommateurs appelés "Cloud consumers". Un Cloud consumer peut être une personne ou une organisation ayant une relation métier avec un ou plusieurs Cloud providers et qui utilise un ou plusieurs de leurs services.

La complexité de ce contexte a fait apparaitre un nouveau type d'entité appelé "Cloud broker" (voir Figure 5.1). Il s'agit d'une entité intermédiaire entre les consommateurs et les fournisseurs de Cloud. Cette entité gère l'utilisation, la performance et la livraison

des services dans un environnement de Cloud computing ainsi que la négociation des relations entre les consommateurs et les fournisseurs. Aujourd'hui, les Cloud brokers sont

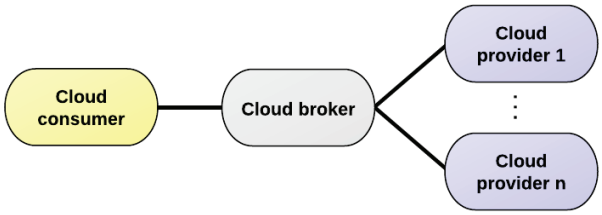


FIGURE 5.1 – Cloud broker

conçus et déployés à la base pour fournir les services et les fonctionnalités souhaitées aux consommateurs et aux fournisseurs de Cloud. Après leurs déploiement et la mise en place d'infrastructures distribuées faisant intervenir plusieurs types de partenaires, les concepteurs se rendent compte que la facturation des services n'a pas été intégrée nativement dans le Cloud broker. En effet, un Cloud broker a besoin d'un certain nombre d'informations pour facturer les consommateurs qui achètent et téléchargent des applications ou qui utilisent des services, des plateformes ou des infrastructures à la demande ... Ces informations sont principalement la provenance de l'application ou du service utilisé, ses coûts de base et ses coûts spécifiques. Nous proposons de modéliser les informations de provenance relatives à la facturation au niveau du Cloud broker en utilisant notre approche de modélisation. Nous avons besoin de modéliser différents types de ressources disponibles chez un fournisseur de Cloud (service, plateforme ou infrastructure), de leurs rattacher des informations de facturation selon leurs caractéristiques (localisation, SLA...) et de définir des coûts qui correspondent aux différentes actions que les consommateurs peuvent effectuer. Des données de provenance relatives à la localisation du fournisseur et du consommateur doivent être aussi intégrées dans ce modèle. Nous avons spécialisé le $MDM$ pour créer un modèle de domaine $MD_{CloudBroker}$ qui regroupe les concepts et les propriétés de ce domaine métier (voir Figure 5.2). Ce modèle définit principalement :

1. Les concepts *Cloud_Consumer* et *Cloud_Provider* qui spécialisent le concept *Actor*. Ces deux concepts définissent le type de l'utilisateur dans un environnement Cloud. Un *Cloud_Consumer* est un consommateur d'une ressource dans le Cloud (un utilisateur final, une application, un développeur ...). Un *Cloud_Provider* est un fournisseur de services dans le Cloud (un hébergeur, une agence Web ...).
2. Le concept *Cloud_Ressource* qui spécialise le concept Document. Il définit le type de la ressource utilisée (CRM, ERP, base de données, plateforme de développement, espace de stockage, capacité de calcul...) et le mode de calcul du coût d'utilisation de cette ressource selon son type de SLA et son emplacement,
3. Le concept *Cloud_Specific_Action* qui spécialise le concept *Action* et définit plusieurs types d'actions : *Deploy_Resource*, *Purchase_Resource*, *Rent_Resource* et *Cus-*

*tomize_Resource*. Chaque type d'action possède une règle pour le calcul du coût de consommation d'une ressource.

A travers cette modélisation, nous avons mis en place un modèle de provenance simple pour la facturation dans un environnement de Cloud computing. La généricité de notre modèle qui se base sur la puissance du modèle OPM est ainsi vérifiée. Effectivement, OPM a fait ses preuves dans plusieurs contextes, spécifiques à la provenance [Chebotko *et al.*, 2010], [Moreau *et al.*, 2010b] ou à d'autres domaines [Li et Boucelma, 2011], [Groth et Moreau, 2011]. En se basant sur ce modèle, il devient possible d'identifier le coût global de l'utilisation d'une ressource à un moment précis et de pouvoir effectuer facilement la facturation de l'utilisation des ressources gérées par un Cloud broker.

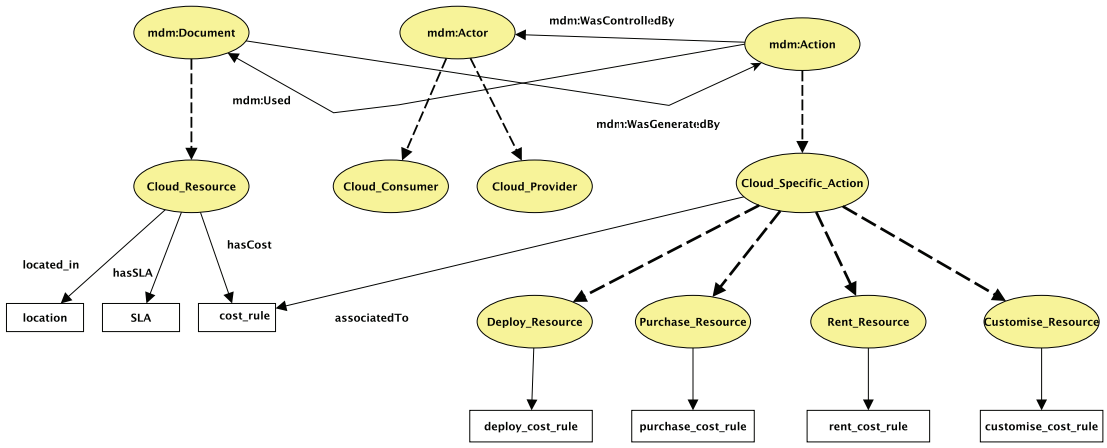


FIGURE 5.2 – Modèle de domaine $MD_{CloudBroker}$ associé à la facturation

## 5.3 Cas d'étude Novapost

### 5.3.1 Description du contexte de validation

La société Novapost, créée en 2007 a mis en place le premier service de dématérialisation opéré sur un mode 100% SaaS (Software As a Service). Ce service, issu d'un partenariat stratégique entre les sociétés Novapost, Acti [1] et IBM (Novapost est IBM Business Partner et membre du club alliance d'IBM), repose sur des solutions développées avec des exigences fortes de sécurité, de performance, de qualité et interopérabilité. Novapost et ses partenaires souhaitent mettre en œuvre un service sur un périmètre de clients restreint aux banques et grands comptes à travers des offres dédiées.

Novapost développe et commercialise des services de gestion documentaire reposant sur une suite logicielle propriétaire, Novapost LX. Cette suite s'articule autour de modules logiciels permettant d'assurer la collecte le routage, l'archivage et la distribution des

1. www.actinet.fr

documents sortants de l'entreprise. Parfaitement complémentaire aux solutions de génération des fichiers (moteurs de composition, etc.), la solution Novapost LX permet de couvrir l'ensemble du processus de dématérialisation des documents (de la génération à partir de données brutes jusqu'à la distribution via une interface Web riche).

Grâce à son architecture orienté service (voir Figure 5.3), la solution répond à des contraintes fortes d'interopérabilité, d'évolutivité et de performance. LX est une solution hautement sécurisée, non intrusive, à capacité de traitement industrielle. Elle est composée des briques logicielles suivantes :

- LX Connect (collecte et interfaçage) : un connecteur logiciel permettant d'intégrer des données et de collecter des documents sur des systèmes d'information distribués, avec une capacité de traitement industrielle et des exigences de sécurité et d'intégrité maximales.
- LX Flow (traitement et gestion de flux) : un moteur de workflow technique orchestrant des chaînes de traitement documentaire.
- LX Delivery (distribution) : une passerelle de distribution vers des portails Internet, des services d'éditique ou de courrier hybride.
- LX SAE : système de stockage et d'archivage électronique assurant l'intégrité, la pérennité, la lisibilité et la force probante des données sur de longues durées.
- LX Live : une solution de portails, de widgets et de Web-services permettant la consultation et la gestion des documents à l'aide d'outils ergonomiques et intuitifs.

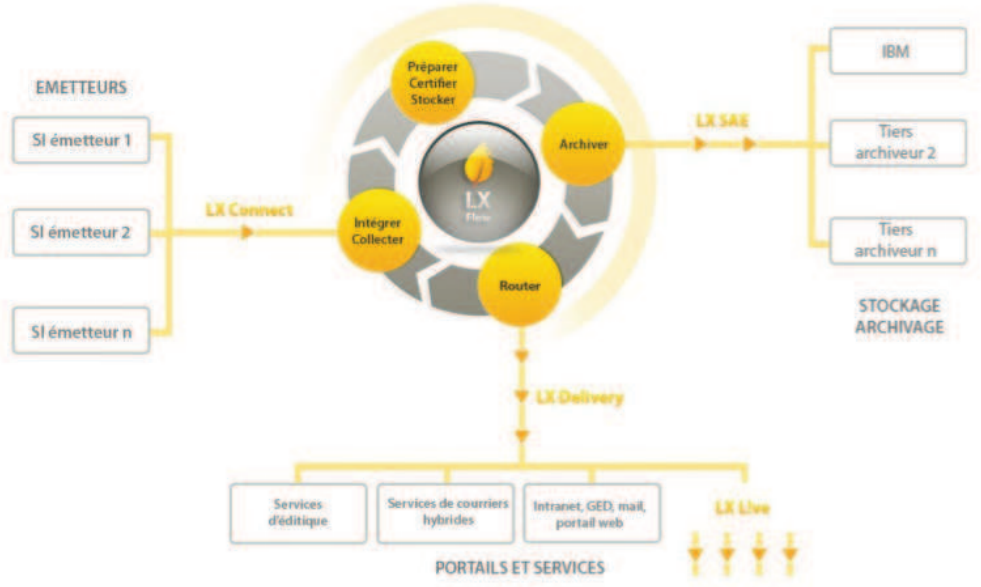


FIGURE 5.3 – Les modules de la solution Novapost

Novapost souhaite développer des services innovants permettant à ses clients une maitrise complète du cycle de vie de leurs documents. Ces services visent à consolider les

offres de dématérialisation existantes en apportant une solution globale combinant des briques technologiques à forte valeur ajoutée.

Pour cela, Novapost considère la provenance comme une méta-donnée de premier ordre sur les documents gérés et archivés. Il s'agit de fournir avec un niveau de granularité fin les détails des opérations effectuées sur les documents. Ce type d'information servira pour effectuer des vérifications précises sur des processus de distribution électroniques et de servir comme preuve numérique en cas de conflit entre un fournisseur et des utilisateurs du service. Ces méta-données doivent être gérées conjointement aux documents et archivées de façon pérenne. C'est dans ce contexte et dans cet objectif que tous nos travaux ont été menés au sein de Novapost sur les problématiques de valeur probante, de traçabilité et de gestion de provenance des documents.

### 5.3.2 Exemples de sources de provenance chez Novapost

Notre besoin principal est de collecter un ensemble de données de provenance décrivant de façon très précise les opérations effectuées sur les documents et les acteurs qui les ont réalisé et de pouvoir ensuite les exploiter. Il s'agit donc de collecter les données de provenance, de les archiver, de les lier aux documents et de pouvoir les interroger. Dans notre travail, nous considérons deux types de sources de provenance : les sources existantes constituées par les logs de serveurs Apache et Tomcat sur lesquels les services sont déployés et les sources spécifiques qui sont issues de services de provenance dédiés.

Le premier type de sources possède l'avantage que les données existent déjà. Elles sont gérées et stockées par les serveurs et ne nécessitent pas une modification des applications et services déjà déployés (voir Figure 5.1). Ces données ne sont pas très riches d'un point de vue sémantique. Cependant, il est possible que les applications et les services soient conçus pour générer des logs plus riches contenant des détails métier.

Le deuxième type est celui des sources contenant des données de provenance générées par des services de provenance dédiés. Ces services génèrent la provenance selon une structure de donnée prédéfinie, dans un format spécifique et les stockent dans un emplacement dédié (base de données, système de fichier ...). Ce type de source possède l'avantage d'être plus riche sémantiquement que le premier mais nécessite un effort d'intégration puisque les applications doivent être modifiées pour intégrer une API appelant ce service. Ce type de source nécessite aussi une gestion de stockage dédiée.

```
1 INFO: Jk running ID=0 time=0/19  config=null
2 8 december. 2010 14:39:31 org.apache.catalina.startup.Catalina start
3 INFO: Server startup in 3485 ms
4 14:39:39 ************   getPrintAndErrorFileNamesByService **************
5 14:39:39 entityType : sender
6 14:39:39 entityId : CHRON01
7 14:39:39 packageRef : CHRON01AA10000329230
8 14:39:39 organizationGroupId : 1
9 14:39:39 organizationId : -1
```

```
10 14:39:39 rootTempDir /var/repository_test/datastore-1/store/tmp
11 14:39:39 service 15
12 14:39:39 ************************
13 14:39:54 {document_path=/var/repository/datastore-1/archive-1/UBISOFT
       /226/807/862/926/UBISO01AA10001160105/, document_extension=pdf,
       document_filename=UBISO01AA10001160105, document_full_path=/var/repository/
       datastore-1/archive-1/UBISOFT/226/807/862/926/UBISO01AA10001160105/
       UBISO01AA10001160105}
14 in createThumbnail 1
15 -thumbnail 90
16 14:50:12 {document_filename=CEGID01AA10000314177.pdf, document_full_path=/var/
       repository_test/datastore-1/store/CEGID/226/809/220/362/CEGID01AA10000314177
       /CEGID01AA10000314177.pdf}
17 14:50:59 ======>Downloading File<======
18 14:50:59 [accountId]:226808419023
19 14:50:59 [spaceId]:22
20 14:50:59 [userId]:126808428406
21 14:50:59 [References]: [CHRON01AA10001169411, CHRON01AA10001178617,
       CHRON01AA10001178618, CHRON01AA10001178619, CHRON01AA10001178620,
       CHRON01AA10001178621, CHRON01AA10001178622, CHRON01AA10001178623,
       CHRON01AA10001178624, CHRON01AA10001227911, CHRON01AA10001227912,
       CHRON01AA10001227913, CHRON01AA10001227914, CHRON01AA10001227915,
       CHRON01AA10001227916, CHRON01AA10001227917, CHRON01AA10001227918,
       CHRON01AA10001227919]
22 14:50:59 ======>Starting download<======
23 14:50:59 [document_full_path]:/var/repository/datastore-1/archive-1/CHRONO
       /226/808/419/023/CHRON01AA10001169411/CHRON01AA10001169411.pdf
24 14:51:00 ======>Finished download<======
```

Exemple de code 5.1: Exemple des données de Log Tomcat

### 5.3.3 Des services REST pour la génération des données de provenance de documents

Au début de notre thèse, nous avons développé un service de traçabilité (appelé LX Track) qui sera utilisé par l'ensemble des modules de la solution Novapost. Il est composé par un ensemble de Web services REST qui ont été développé en se basant sur le framework Jersey [2]. Nous avons développé aussi un client Java pour ce service qui permet d'interagir avec les différents modules de la solution Novapost.

Ces Web services permettent de créer des traces pour plusieurs types d'actions sur les documents, de les lier aux documents, de les enrichir, de les consulter et de faire des recherches selon plusieurs critères. Dans cette première version (développée avant la conception de notre framework sémantique) et pour des raisons de pérennité et de standardisation, nous avons choisi de définir une structure de trace en XML. Les Web services que nous avons développés permettent ainsi de générer des données de provenance dans un format XML et de les stocker. Aussi, des méta-données sur ces données permettant de les lier aux documents sources sont générées et stockées dans une base MySQL. L'architecture du service LX Track que nous avons développé est présentée sur la Figure 5.4 ci-dessous.

2. http ://jersey.java.net

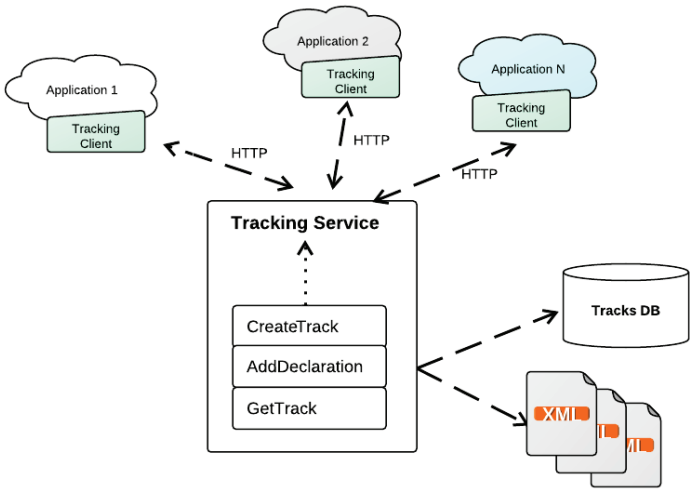


FIGURE 5.4 – Architecture du service de traçabilité développé

#### 5.3.3.1 Web service pour la création de traces

Ce Web service permet à une application de déclarer la création d'une trace associée à un document. Il crée une entrée en base de données permettant d'associer l'identifiant d'une application et d'un document à un identifiant unique appelé TUUID (Track Universal Unique Identifier). L'identifiant d'application est un identifiant du service ou du module qui utilise le client pour interagir avec le service de traçabilité. Si la création de la trace est effectuée avec succès, le Web service retourne le TUUID associé. L'exemple ci-dessous (voir Exemple de code 5.2) illustre la spécification de ce Web service et un exemple d'instanciation de ces paramètres.

```
1 Action : POST
2 Ressource : /api/tracks/
3 {
4 ``applicationId'': ``{Novapost}'',
5 ``documentId'': ``{paie02-02 2011}'',
6 ``fingerPrint'':``{6198e34ab1a265319ba1b1dded36f2867eb9b063dcff197b}'',
7 ``algorithm'': ``{SHA-256}''
8 }
9 applicationId : chaîne obligatoire, non vide et ne contenant pas d'espaces.
10 documentId : chaîne obligatoire non vide.
11 fingerPrint: chaîne obligatoire non vide.
12 algorithm: chaîne obligatoire non vide.
13
14 Réponse :
15 201 - Created { ``tuuid'': 967199433184360762306350718947681743631166899}
16 400 - Bad request : paramètres invalides ou manquants.
17 409 - Conflict: trace déjà existante pour le couple {applicationId,documentId}
18 500 - Internal error
```

Exemple de code 5.2: Service de création de traces

#### 5.3.3.2 Web service pour la déclaration d'une action (Archivage)

Ce Web service permet de déclarer une action d'archivage sur un document donné. La trace qui existe déjà pour ce document est enrichie avec la déclaration d'archivage qui contient les paramètres associés à la stratégie d'archivage utilisée. L'exemple de code suivant (voir Exemple de code 5.3) présente ce Web service et ses paramètres.

```
1 Action : PUT
2 Ressource : /api/tracks/{applicationId}/{documentId}/archive
3 {
4 ''archiverId'': ''{CDC}'',
5 ''externalArchiveId'': ''{paie2011.pdf}'',
6 ''status'': ''{ARCHIVED_PENDING}''
7 }
8 archiverId : chaîne obligatoire, non vide et ne contenant pas d'espaces.
9 externalArchiveId : chaîne obligatoire non vide.
10
11 Réponse :
12 204 - No Content : requête traitée avec succès mais pas d'information à renvoyer
13 400 - Bad request : paramètres invalides ou manquants
14 404 - Not found: cette trace n'existe pas
15 500 - Internal error
```

Exemple de code 5.3: Service d'ajout de déclaration d'une action d'archivage

#### 5.3.3.3 Web service pour la récupération de la trace complète d'un document

Ce Web service permet de récupérer la trace XML complète relative au cycle de vie d'un document identifié par le couple {applicationId,documentId}. Même si le document possède plusieurs identifiants associés à plusieurs application, il est possible de récupérer cette même trace avec n'importe quel couple {applicationId,documentId} associé au document. L'exemple suivant (voir Exemple de code 5.4) présente la description de ce Web service.

```
1 Action : GET
2 Ressource : /api/tracks/{applicationId}/{documentId}
3
4 Reponse :
5 200 - OK (Content-Type: application/xml)
6 404 - Not found - pas de trace associée au couple {applicationId, documentId}
7 500 - internal error
```

Exemple de code 5.4: Service de récupération d'un trace par identifiant d'application et de document

```
1 <Track tuuid="967199433184360762306350718947681743 63-b2e6a99f-b0dd-4eea-b278-58
      d493d4a15f" applicationReference="LX-SAE" trackCreationDate="2010-05-07
      10:38:41" language="en-US">
2  <Headers>
3    <Document reference="CEGID01AA10000314177" name="CEGID01AA10000314177.pdf" />
4    <FingerPrints>
5      <FingerPrint value="9
           DF18DD30BBC482CACC72B78359F86AB337073621C37AC7B4F33F7850CD0200B" algorithm
           ="SHA-256" />
```

```
6       </FingerPrints>
7     </Headers>
8     <Parents />
9     <Declarations>
10      <Declaration declarationTimeStamp="2010-05-07 10:40:16">
11        <UnsignedBloc>
12          <Action name="ARCHIVE" applicationReference="LX-SAE"  location="IBM-
                MONTPELLIER" path="/var/repository_test/datastore-1/store/CEGID
                /226/809/220/362/CEGID01AA10000314177" />
13        </UnsignedBloc>
14      </Declaration>
15      </Track>
```

Exemple de code 5.5: Exemple de trace XML pour une action d'archivage

### 5.3.4 Intégration du service de traçabilité développé dans notre architecture de PMS

Nous avons développé ces services pour générer des traces avant d'avoir fini la définition complète de notre architecture de PMS. Cependant, notre API était conçue de façon générique séparant le modèle métier de la couche Web service. Pour cette raison, elle peut s'intégrer dans notre architecture si l'on développe un « wrapper » spécifique permettant de générer directement la provenance selon le modèle souhaité et de la stoker dans le module de stockage du PMS. Les utilisateurs souhaitant ensuite exploiter ces données peuvent dans ce cas passer par le module d'interrogation pour effectuer leurs requêtes.

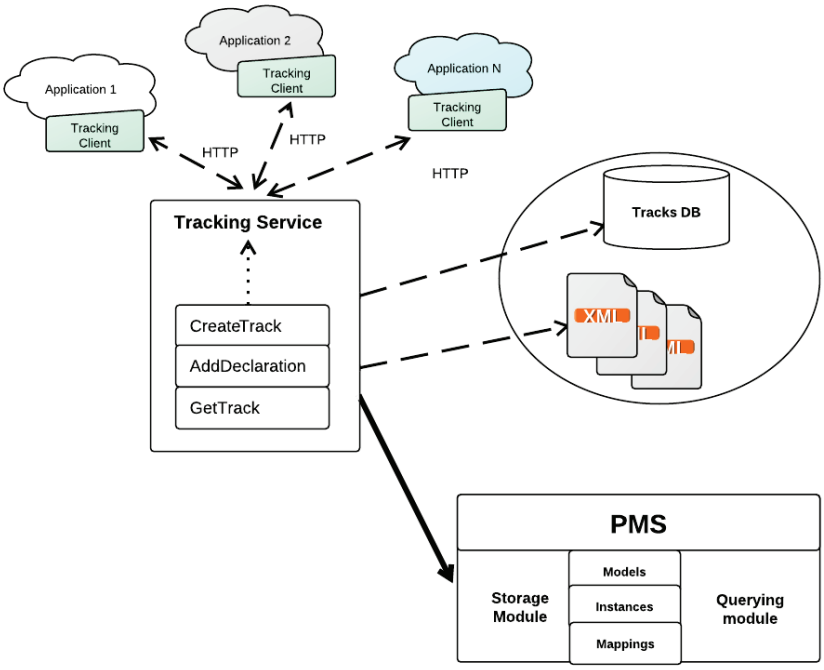


FIGURE 5.5 – Intégration du service de traçabilité dans notre architecture de PMS

### 5.3.5 Synthèse

Nous avons présenté dans cette partie la solution Novapost et ses différents modules. Ensuite, nous avons présenté des exemples de données de provenance qui étaient générées automatiquement (log du serveur tomcat) ou via des services dédiés (services REST pour la génération de traces XML). Ces services assurent la génération, l'enrichissement et la consultation de traces XML et ont été développés avant la mise en place complète de notre framework. L'utilisation du format XML nous garantit une pérennité de stockage. Par contre elle limite les capacités d'interrogation de ces données. Aussi, ces services nécessitent une intégration du client dans tous les modules qui effectuent des actions sur les documents ce qui exige une bonne connaissance de leurs spécificités. Étant donné que l'API que nous avons développé pour ces services est indépendante du modèle de stockage, nous avons proposé une solution assurant son interfaçage avec un PMS.

Cette première solution nous a permis de bien comprendre la spécificité et la variété de l'information de provenance et nous a montré l'importance d'être capable de paramétrer la structure de la trace par une modélisation métier spécifique.

## 5.4 Validation non-fonctionnelle

Dans cette partie, nous proposons d'instancier l'architecture logique de PMS proposée dans le chapitre 4 et de l'évaluer. Nous avons choisi deux technologies pour le module de stockage et nous discutons la modélisation, le chargement et l'interrogation de la provenance pour chacune d'entre elles. La première technologie de stockage utilisée appartient à la famille des triple stores RDF. Ce type de technologie permet de stocker les données dans des formats sémantiques et non pas relationnels. La majorité des systèmes de cette famille supportent nativement l'interrogation avec un langage de requêtes sémantique comme RQL ou SPARQL [Bailey *et al.*, 2005]. L' utilisation d'un RDF store s'inscrit dans la continuité de notre approche qui se base sur l'utilisation des technologies du Web sémantique pour la gestion de provenance. Nous avons choisi Sesame comme RDF store pour le stockage de la provenance. Les données stockées seront par la suite interrogées avec des requêtes SPARQL.

La deuxième technologie de stockage que nous proposons d'utiliser est un SGBD NoSQL couplé avec une interface de requêtes basée sur map/reduce. Contrairement aux SGBDs relationnels, les systèmes NoSQL n'imposent pas la notion de schéma sur les tables. Aussi, ils possèdent l'avantage d'être scalable en augmentant le nombre de serveurs. Nous avons choisi CouchDB qui est une base de données NoSQL orientée documents et qui propose une interface de requêtes map/reduce. Par contre, nous ne pourrons pas dans ce cas utiliser les mêmes requêtes SPARQL définies pour Sesame et nous devons les traduire en map/reduce.

### 5.4.1 Description des données et de l'environnement de test

Notre travail d'expérimentation a été réalisé en prenant deux sources de provenance chez Novapost : le log d'un serveur Tomcat [3] ayant 850 MB de taille et le log d'un serveur Web Apache [4] ayant 250 MB de taille. Ces deux sources de provenance contiennent les traces de traitement, de distribution et d'archivage de documents sur quatre mois pour 20 entreprises clientes et un total d'un million de documents pdf (des bulletins de salaire) environ. Ces sources de provenance contiennent approximativement 2.5 million de lignes de logs qui doivent être analysées et intégrées dans le module de stockage utilisé.

Nous avons réalisé nos expérimentations sur un serveur DELL ayant la configuration suivante : processeur Intel Quad-Core Xeon W3530 2.8Ghz, Mémoire vive : 12 GB DDR2 667, disques durs 220 GB (10,000 rpm) SATA2. Le système d'exploitation installé est Fedora 13 (Goddard) 64-bit. Nous avons installé les logiciels suivants sur ce serveur :

1. Apache Tomcat (version 6.0.35),
2. Sesame [5] (version 2.6.3). Cette version est aussi caractérisée par son support au langage SPARQL 1.1. Nous avons déployé Sesame comme une application Web sur le serveur tomcat que nous avons installé. Sur ce serveur, nous avons créé un repository en stockage natif (native storage),
3. Java (JRE v1.6.0 18),
4. CouchDB 1.1.0,
5. BigCouch 0.4a (pour gérer la scalabilité).

### 5.4.2 Utilisation d'un RDF store pour le stockage de la provenance : Sesame

Un triple store RDF est un système qui fournit un mécanisme pour le stockage et l'accès aux données RDF. Nous avons choisi d'utiliser Sesame qui est un framework open source pour le stockage et l'interrogation des données RDF et RDFS. Ce framework a été initié par Aduna [6] comme une partie du projet On-To-Knowledge [7] pour devenir ensuite un projet open source portant comme nom openRDF [8].

Sesame supporte plusieurs types de stockage, chacun ayant des caractéristiques spécifiques. Ces types sont les suivants :

1. Stockage relationnel (RDMS backend) : il se base sur un modèle de stockage relationnel vertical où chaque déclaration RDF est stockée comme un tuple unique dans une table. Des optimisations et des mappings (comme les mappings des URIs et des chaines de caractères à des identifiants de type entier) sont définis pour de meilleures

3. http ://tomcat.apache.org
4. http ://httpd.apache.org
5. www.openrdf.org
6. www.aduna-software.com
7. On-To-Knowledge (IST-1999-10132). voir http ://www.ontotext.com/research/otk
8. www.openRDF.org

performances et pour minimiser l'espace disque requis. Ce type de stockage supporte MySQL, PostgreSQL et Microsoft SQL server.

2. Stockage en mémoire (in-memory backend) : ce type de stockage est facile à installer et à déployer. Il est aussi performant en insertion, inférence et interrogation. Cependant, il possède l'inconvénient d'être très couteux en terme de ressources et non scalable sur de très grand volumes de données.
3. Stockage natif (native storage) : ce type de stockage assure une bonne performance ainsi qu'une meilleure scalabilité que les types précédents. Aussi, il ne nécessite pas d'installer d'autres logiciels tiers (à l'encontre du stockage relationnel). Il se base sur un stockage à plat dans des fichiers et un accès basé sur les classes Java I/O (package java.nio). C'est ce type de stockage que nous avons utilisé par la suite.

#### 5.4.2.1 Représentation de la provenance dans Sesame

Pour sauvegarder la provenance dans Sesame, nous devons utiliser une ontologie permettant de modéliser les données de provenance. Nous utilisons l'ontologie définie dans 3.4 (voir Figure 3.4) qui représente la sémantique métier de Novapost. Une fois cette ontologie instanciée via la fonction d'import, les instances peuvent être stockées dans Sesame.

Nous avons utilisé la librairie python RDFLib [9] pour implémenter la fonction d'import. Cette fonction analyse les logs en entrée et crée des graphes RDF (dans des fichiers .rdf) correspondants au $MD_{Novapost}$. Ces fichiers sont ensuite chargés en utilisant l'invité de commande de Sesame. En moyenne, un document génère sept triplets, une action génère sept triplets et un acteur génère six triplets. L'exemple suivant (voir Exemple de code 5.6) présente des données RDF générées avec notre fonction d'import. Cette fonction nous a généré environ 24 millions de triplets RDF (24.000.416) que nous pouvons interroger avec des requêtes SPARQL.

```
1 <nova:AuthenticatedActor rdf:about=http://www.track.novapost.fr/instantiated/
      actors/sftpSender>
2 <nova:hasId>sftpSender</nova:hasId>
3 <nova:type>AuthenticatedActor</nova:type>
4 <nova:hasActorId>sftpSender</nova:hasActorId>
5 </nova:AuthenticatedActor>
6 <nova:ProbativeValueAction rdf:about=http://www.track.novapost.fr/instantiated/
      actions/send28-11-2011>
7 <nova:hasInput rdf:resource=http://www.track.novapost.fr/instantiated/documents
      /1423438/>
8 <nova:precededBy>emit-63</nova:precededBy>
9 <nova:performedAt rdf:datatype=http://www.w3.org/2001/XMLSchema#dateTime
      >2011-11-28T21:32:52</nova:performedAt>
10 <nova:performedBy>sftpSender</nova:performedBy>
11 <nova:hasActionId>send-28-11-2011</nova:hasActionId>
12 <nova:hasOutput rdf:resource=http://www.track.novapost.fr/instantiated/documents
      /744439/>
```

9. www.rdflib.org

```
13 </nova:ProbativeValueAction>
14 <nova:ProbativeValueDocument rdf:about=http://www.track.novapost.fr/instantiated
       /documents/1423438>
15 <nova:hasApplicationId>sftpSender</nova:hasApplicationId>
16 <nova:hasStatus>sent</nova:hasStatus>
17 <nova:hasDocumentId>doc1423438</nova:hasDocumentId>
18 <nova:hasSystemId>hrAccess</nova:hasSystemId>
19 <nova:hasSHA256FingerPrint>
       D12AB8F6200BD0AAE1B1F5B9B5317F8F4113B2B9C015B3734045FA463B5A6D0D</nova:
       hasSHA256FingerPrint>
20 </nova:ProbativeValueDocument>
```

Exemple de code 5.6: Exemple de données RDF sauvegardées dans Sesame

#### 5.4.2.2 Requêtes SPARQL sur les données de provenance

Après le stockage des données dans Sesame, nous allons les interroger avec SPARQL (2.4.3.5). L'interrogation avec ce langage implique l'utilisation d'ontologies dans l'entête de la requête pour définir le (ou les) modèle(s) d'interrogation. Ces requêtes ciblent des $MDIs$ déjà importés dans le store. L'avantage de SPARQL est son aspect déclaratif qui permet de formuler facilement des requêtes qui portent sur des concepts et des propriétés métier. L'agrégation et la récursivité sont aussi assurées dans SPARQL qui peut résoudre des requêtes de chemin. Les exemples ci-dessous (voir Exemple de code 5.7, Exemple de code 5.8 et Exemple de code 5.9) présentent respectivement les requêtes Q1, Q3 et Q6 que nous avons présentées dans le chapitre 3 formulées avec SPARQL.

```
1 PREFIX nova: <http://track.fr/DM/novapost/probative.owl>
2 SELECT ?documents
3 FROM <http://track.fr/idms/novapost.rdf>
4 WHERE { ?actions nova:performedBy ''nova-archiverAgent''.
5         ?actions nova:hasInput ?documents.
6 }
```

Exemple de code 5.7: Q1 en SPARQL

```
1 PREFIX nova: <http://track.fr/DM/novapost/probative.owl>
2 SELECT (count(?documents) as ?c)
3 FROM <http://track.fr/idms/novapost.rdf>
4 WHERE { ?actions nova:hasInput ''doc02-10-C'' .
5         ?actions nova:hasOutput ?documents .
6         ?documents nova:hasStatus ''archived'' .
7       }
```

Exemple de code 5.8: Q3 en SPARQL

```
1 PREFIX nova: <http://track.novapost.fr/DM/mdm.owl>
2 SELECT ?linkedAction
3 FROM <http://track.fr/idms/novapost.rdf>
4 WHERE { ?action nova:hasOutput ''doc-02-10-C'' .
5         ?action (nova:hasInput/^nova:hasOutput)* ?linkedAction .
6         }
```

Exemple de code 5.9: Q6 en SPARQL

### 5.4.3 Utilisation d'un système NoSQL pour le stockage de provenance : CouchDB

Les bases NoSQL (Not only SQL) [Cattell, 2011] désignent une catégorie de systèmes de gestion de base de données non relationnels, scalables de manière horizontale sur des machines ayant une configuration standard. Bien que ce nom est controversé et que plusieurs variantes de systèmes NoSQL existent, ils possèdent tout ou partie les propriétés suivantes :

- scalabilité horizontale des opérations sur plusieurs machines,
- réplication et partitionnement des données sur plusieurs machines,
- une interface ou un protocole d'appel simple (contrairement à SQL),
- un modèle concurrentiel et transactionnel faible contrairement au modèle ACID de la majorité des bases de données relationnelles,
- utilisation efficace des indexes distribués et de la mémoire RAM pour le stockage des données,
- absence de la notion de schéma qui fixe la structure des données, de nouveaux attributs peuvent être ajoutés dynamiquement aux structures existantes.

Les systèmes NoSQL possèdent plusieurs différences. On trouve les systèmes basés sur le hachage distribué comme Memcached [10], les systèmes clé-valeurs distribués comme BigTable de Google [Chang *et al.*, 2006]. En effet, les systèmes NoSQL pionniers étaient une preuve de concept validant de nouvelles approches pour la scalabilité, la cohérence, le partitionnement et la tolérance aux pannes. Memcached a prouvé que l'utilisation des indexes en mémoire peut être hautement scalable et permet de distribuer et de répliquer les données sur plusieurs nœuds. Dynamo [Strauch, 2010] a prouvé l'utilité du concept de cohérence à terme (Eventual Consistency) comme une solution pour assurer la haute disponibilité et la haute scalabilité : il n'y a pas de garantie que les données récupérées à partir d'un nœud soient à jour mais la garantie de propager ces mises à jour sur tous les nœuds est assurée. D'un autre coté, Google a montré à travers BigTable que le stockage persistant peut être scalable sur des milliers de nœuds.

Plusieurs de ces systèmes sont couplés avec un modèle de parallélisation appelé map/reduce [Dean et Ghemawat, 2010] qui a été inventé par Google et rendu populaire avec son implémentation open source Hadoop [11]. Map/reduce permet d'exécuter des requêtes sur de larges volumes de données en utilisant une méthode de parallélisme simple : map qui distribue une opération sur tous les nœuds et reduce qui agrège les résultats produits sur les différents nœuds.

Si nous catégorisons les systèmes NoSQL, nous identifions les quatre catégories suivantes :

10. www.memcached.org
11. www.hadoop.apache.org

1. les systèmes orientés colonne comme BigTable et Cassandra [Strauch, 2010],
2. les systèmes clé-valeur comme Amazon DynamoDB [Sivasubramanian, 2012] et Project Voldemort [12],
3. les systèmes orientés document (comme CouchDB [13] et MongoDB [14])
4. les systèmes orientés graphes comme (Neo4J [15]).

Grâce à leur utilisation et leur succès chez les grands acteurs du Web comme Google, Facebook ou encore Twitter, les systèmes NoSQL ont dépassé aujourd'hui l'étape de concept et ont fait leurs preuves dans des environnements de production et répondent à de hautes exigences de performance et de scalabilité. Cependant, plusieurs discussions et travaux s'interrogent aujourd'hui sur l'efficacité des bases NoSQL par rapport aux bases de données parallèles [Stonebraker *et al.*, 2010]. Nous jugeons que les deux mondes, relationnel et NoSQL peuvent et doivent co-exister pour la simple raison qu'il répondent à des problématiques différentes. Dans notre contexte de gestion de provenance, nous jugeons que NoSQL est assez prometteur et permettra de proposer des solutions efficaces pour construire des systèmes permettant de stocker et d'interroger avec efficacité de grands volumes de données en se basant uniquement sur des machines ayant une configuration standard [Pavlo *et al.*, 2009].

Plus précisément, nous pensons que l'utilisation d'un système NoSQL pour le stockage de la provenance est une solution intéressante pour les raisons suivantes :

- la structure de la provenance est flexible et varie selon le besoin d'utilisation, le type d'application ... Ce besoin correspond bien à une caractéristique de base des systèmes NoSQL qui ne nécessitent pas de définir un schéma fixant la structure et le type de données,
- la provenance n'a pas besoin de contraintes transactionnelles lorsqu'elle est intégrée ou interrogée,
- la provenance est statique, elle n'est pas mise à jour et est utilisée en mode d'ajout seulement (tel est le cas pour les données analytiques par exemple),
- les applications génèrent de grands volumes de provenance. Ceci nécessite l'utilisation de technologies scalables, extensibles à la demande sur des machines standards,
- les données de provenance nécessitent des techniques d'interrogation efficaces sur de gros volumes de donnés. C'est le cas pour la plupart des bases NoSQL qui proposent l'utilisation des techniques d'interrogation basées sur map/reduce par exemple ce qui permet une sélection et une agrégation des données efficace grâce au parallélisme.

12. http ://project-voldemort.com
13. http ://www.couchdb.apache.org
14. http ://www.mongodb.org
15. http ://www.neo4j.org

#### 5.4.3.1 Représentation et interrogation de la provenance dans CouchDB

Nous avons choisi d'utiliser comme base NoSQL CouchDB qui est une base de données non relationnelle orientée documents. CouchDB a été développé en Erlang [16] et utilise plusieurs technologies conçues pour le Web comme les services REST [Fielding, 2000] pour les opérations d'insertion, mise à jour, lecture et suppression et JSON comme format de stockage de données. CouchDB introduit la notion de document qui est la structure équivalente à un tuple dans un SGBD relationnel. L'interrogation d'une base CouchDB est effectuée à travers des vues dont la définition et l'exécution se base sur le modèle map/reduce. CouchDB permet aussi de répliquer les données sur plusieurs nœuds de façon incrémentale.

Nous proposons de représenter les instances d'un $MDI$ en se basant sur trois types de documents (document au sens de CouchDB) que nous définissons. Ces types sont *ActorDocument*, *ActionDocument* et *Document*. Nous proposons aussi de matérialiser la corrélation entre les artifacts en utilisant un type de document dédié que nous appelons *CorrelationDocument* qui contient un ensemble d'informations permettant de définir les liens entre les identifiants d'objets distribués. Cette modélisation permet ainsi de projeter les concepts du $MDM$ dans CouchDB. N'importe quel type de document parmi les types que nous avons défini peut référencer d'autres types de documents. Ainsi, notre modélisation permet de manipuler les concepts du $MDM$ comme des documents et donc comme des objets de premier ordre.

Nous matérialisons les liens qui existaient dans les triplets RDF en tant qu'attributs dans nos différents documents. Nous présentons sur la Figure 5.6 illustrant un exemple de notre modélisation de la provenance dans CouchDB. Cet exemple présente des documents et les liens qui les relient. Sur cet exemple, l'action "send-28-11-2011" est précédée par l'action "emit-63". Cette action a pris en entrée le document identifié par "744438" pour générer le document identifié par "744439". Elle a été effectuée par l'acteur "sftpSender". Nous illustrons aussi une relation de correspondance entre le document identifié par "744439" et celui identifié par "cust356_12". Cette correspondance est sauvegardée dans un document spécifique "corr_569".

Certes, ce choix de modélisation de la provenance dans CouchDB n'est pas unique et d'autres alternatives pour la modélisation des documents dans CouchDB existent. Une première alternative est de représenter et de stocker dans un document unique toute la provenance d'une source donnée. L'inconvénient dans ce cas est de dépasser la taille maximale autorisée pour un document dans CouchDB (fixée à 4GB). Une autre alternative est de représenter dans un document unique toutes les informations relatives à une déclaration de provenance (qui correspond à une ligne dans un fichier de log par exemple). Dans ce cas, toutes les informations relatives à un action sont stockées dans ce même document en tant qu'attributs. Une telle représentation est pauvre d'un point de

16. www.erlang.org

vue sémantique et est centrée uniquement sur les processus (i.e les actions). Elle permet de créer uniquement des liens entre les différentes actions. Par contre, les liens entre les documents et les acteurs seront définis comme des attributs de ce même document ce qui rend la formalisation des requêtes map/reduce complexes. Une troisième alternative est d'utiliser une modélisation similaire à celle que nous avons choisi (c'est à dire de définir trois types de documents et de les référencer par des attributs) en dupliquant les références entre les documents dans les deux sens. Cette modélisation facilite la navigation dans le graphe de provenance à partir de n'importe quel document.

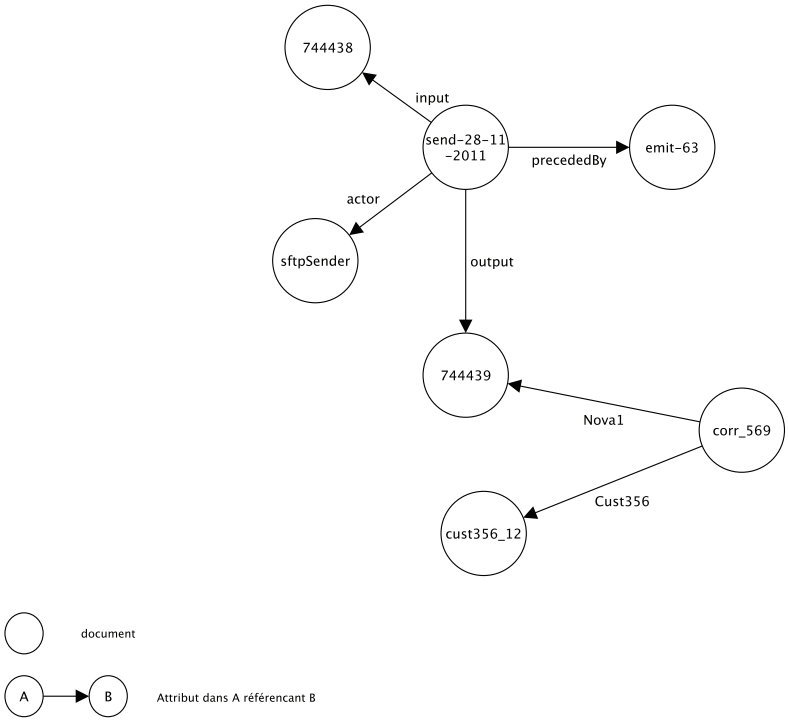


FIGURE 5.6 – Modélisation de la provenance dans CouchDB

Étant donné qu'il s'agit d'une base NoSQL qui ne possède pas un schéma prédéfini, les instances du $MDM$ peuvent être enrichies pour créer dans le même document un modèle de domaine instancié. Ainsi, nous n'avons pas besoin de répliquer autant de fois les $MDIs$ à chaque fois que nous souhaitons instancier un nouveau $MD$. Il suffit juste d'enrichir le $MDI$ existant. L'exemple de code suivant (voir Exemple de code 5.10) illustre une représentation JSON de données de provenance stockées dans CouchDB.

```
1 Action Document:
2 {
3     "_id": "001e38d82a5818ea60dee3cef1001525",
4     "_rev": "1-0d7086883d47c872d5c5376113d11ec4",
5     "name": "SEND",
6     "id": "send-28-11-2011",
7     "type": "ActionDocument",
8     "timestamp": "28/11/2011-21:32:52",
9     "input": [
```

```
10          "744438"
11      ],
12      "output": [
13          "744439"
14      ],
15      "actor": "sftpSender",
16      "precededBy": "emit-63"
17  }
18
19  Actor Document:
20  {
21      "_id": "001e38d82a5818ea60dee3cef103794c",
22      "_rev": "1-702717b4f4266998791c637abdd6ea4b",
23      "name": "sftpSender",
24      "type": "ActorDocument",
25      "characteristic": "Authenticated"
26  }
27
28  Document:
29  {
30      "_id": "001e38d82a5818ea60dee3cef1032d1c",
31      "_rev": "1-2dee8a50897071be3140dabac4290233",
32      "type": "Document",
33      "status": "sent",
34      "ids": [ "cegid:744438","nova:paie\_11-2011-custCE4"
35      ],
36      "path":[''output,split,744439,28/11/2011-21:32:55'' ],
37      "sha256": "D12AB8F6200BD0AAE1B1F5B9B5317F8F4113B2B9C015B3734045FA463B5A6D0D",
38      "identifier": "744438"
39  }
```

Exemple de code 5.10: Représentation de la provenance en JSON dans CouchDB

Nous avons développé la fonction d'import en Java et nous avons utilisé le client Ektrop [17] qui est un client Java proposant des fonctions de persistance pour interagir avec les bases CouchDB. Comme la logique orientée triplet de Sesame est différente de la logique orientée document de CouchDB, le nombre d'enregistrements générés par la fonction d'import sur les mêmes logs était différent du nombre de triplets RDF générés. En effet, notre fonction d'import a généré 3.000.517 documents. Pour les charger dans notre module de stockage, nous avons utilisé la fonction « bulk insert » du client Ektrop. Cette fonction est efficace pour une intégration de documents en masse dans CouchDB.

Pour pouvoir interroger les données de provenance stockées dans CouchDB, nous avons besoin de créer des vues. Elles peuvent être définies en Javascript. Dans CouchDB, deux types de vues existent :

- les vues temporaires : elles ne sont pas matérialisées en base et sont exécutées à la demande. L'exécution de ce type de vue est lente. Cette lenteur augmente au fur et à mesure que la taille de la base augmente.
- les vues permanentes : ce type de vue est stocké dans un type spécial de document appelé design document. Un design document peut contenir plusieurs vues. Chaque vue est identifié par un nom unique au sein de ce document. Elle définit nécessairement une fonction map et optionnellement une fonction reduce.

17. http ://ektorp.org

La fonction map consulte tous les documents dans CouchDB de façon incrémentale pour générer un premier résultat. Ce résultat est une liste ordonnée de clés/valeurs. Elles sont définies par l'utilisateur qui développe cette fonction. Cette fonction fait appel à la fonction intégrée *emit(clé/valeur)* de CouchDB entre 0 et N fois par document pour ajouter une entrée dans le résultat de map. Les exemples de code suivants (voir Exemple de code 5.11, Exemple de code 5.12 et Exemple de code 5.13) présentent nos requêtes Q1, Q2 et Q6 en map/reduce.

```
1   ``documents_by_actor": {
2             ``map": "function(doc)
3                {if((doc.actor) && (doc.type==`ActionDocument'))
4                    emit(doc.actor, doc._id)}''
5        }
```

Exemple de code 5.11: Q1 en map/reduce

```
1 ``documents_archived_number": {
2            "map": "function(doc) {if((doc.type=='Document') && (doc.status) && (doc.
                 archiver) &&(doc.identifier))
3                   emit([doc.identifier,doc.status,doc.archiver], doc._id)}",
4             "reduce": ``_count"
5        }
```

Exemple de code 5.12: Q3 en map/reduce

```
1
2 ``path_by_docid": {
3 if (doc.identifier) {
4     emit([doc.identifier, 0], doc.path);
5     if (doc.inputs) {
6       for (var i in doc.inputs) {
7         emit([doc.identifier, Number(i)+1], {path: doc.path[i]});
8       }
9   }
10 if (doc.outputs) {
11       for (var j in doc.outputs) {
12         emit([doc.identifier, Number(j)+1], {path: doc.path[j]});
13       }
14     }
15   }
16 }
```

Exemple de code 5.13: Q6 en map/reduce

Ces exemples montrent que même si les requêtes map/reduce ne sont pas très complexes à définir et à comprendre, elles ne sont pas aussi déclaratives et lisibles que les requêtes SPARQL. La complexité de ces fonctions augmente d'autant que la requête est complexe.


```
1 {"id":"8447c2517d6bbf56b02523c3a9d74c6e","key":{"identifier":"UBISOFT:","localId
      ":"","identifcationSystemId":"UBISOFT"},"value":"8447c2517d6bbf56b02523c3a9d74c6e
      "},
2 {"id":"84c9764b26b5951e57f1fda6542ca8aa","key":{"identifier":"UBISOFT:","localId
      ":"","identifcationSystemId":"UBISOFT"},"value":"84c9764b26b5951e57f1fda6542ca8aa
      "},
```

```
3 {"id":"8659596bc94d33528c37b472dcfeab10","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"8659596bc94d33528c37b472dcfeab10
      "},
4 {"id":"95ddc9f06131f02e762230658b68cab4","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"95ddc9f06131f02e762230658b68cab4
      "},
5 {"id":"98ac0810d6a85438dc776d06b87b9b18","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"98ac0810d6a85438dc776d06b87b9b18
      "},
6 {"id":"a01d8a53ca64d37279dfb0f5a6002445","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"a01d8a53ca64d37279dfb0f5a6002445
      "},
7 {"id":"b36addfb79f83fa4b7eea53c5705c65a","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"b36addfb79f83fa4b7eea53c5705c65a
      "},
8 {"id":"cdb8d8ad9fdcd42be4296c4532d87e46","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"cdb8d8ad9fdcd42be4296c4532d87e46
      "},
9 {"id":"dccdba34a36fcb5cb26231c3747cfb49","key":{"identifier":"UBISOFT:","localId
      ":""},"identifcationSystemId":"UBISOFT"},"value":"dccdba34a36fcb5cb26231c3747cfb49
      "},
```


Exemple de code 5.14: Un exemple du résultat de l'exécution de la fonction map pour Q1

#### 5.4.3.2 Expérimentations et analyse des résultats

**Chargement des données de provenance**

Comme la structure de données de provenance dans Sesame n'est pas identique à celle de CouchDB, le nombre d'enregistrement généré pour les mêmes sources de provenance est différent. La fonction d'import dans Sesame a généré 24.000.416 de triplets RDF, celle de CouchDB a généré 3.000.517 documents. La table 5.1 ci-dessous donne le temps d'insertion requis pour charger les données de provenance dans les deux SGBD. Sur ce tableau, nous utilisons l'équivalent de lignes de log insérées car les structures de données et le nombre d'enregistrement sont différents pour les deux systèmes. Pour des ensembles de données de taille inférieure à un million de triplet, Sesame était rapide et le temps de chargement était raisonnable. Pour des ensembles de plus grande taille, le temps de chargement nécessaire devient beaucoup plus important et n'est plus linéaire. Pour charger l'ensemble des triplets générés (24.000.416), Sesame a passé plus que 7 heures (28302 secondes). Pour CouchDB, nous avons utilisé la fonction de chargement massif (bulk insert) qui nous a permis de charger les données d'une manière efficace et dans un temps presque linéaire relativement au nombre de documents (voir Table 5.1 et Figure 5.7).

Pour CouchDB, un temps pour le calcul des vues matérialisées avant l'exécution des requêtes est requis. Nous présentons dans le tableau ci-dessous (voir Table 5.2) le temps nécessaire pour deux ensembles de 1 et 6 millions de documents. Ce temps est linéaire. Il dépend du volume de données et du type de la vue (simple ou paramétrée). Le fait d'ajouter de nouvelles données à des données déjà indexées nécessite un temps supplémentaire pour recalculer la vue matérialisée. Ce temps dépend également du volume de données ajoutées.

**Interrogation des données de provenance**

| Nombre de lignes de log (K) | 50 | 100 | 500 | 1000 | 2500 |
|---|---|---|---|---|---|
| Temps de chargement (Sesame) | 2.97 | 135 | 1948 | 4080 | 28302 |
| Temps de chargement (CouchDB) | 20.89 | 43.36 | 219 | 427 | 1044 |

TABLE 5.1: Temps de chargement de la provenance dans Sesame et CouchDB (en secondes).

| Nombre de documents (M) | 1 | 6 |
|---|---|---|
| Temps de calcul des vues | 175 | 363 |
| Temps de calcul des vues (ajout de + 10 % ) | 26 | 41 |

TABLE 5.2: Temps de calcul des vues matérialisées pour CouchDB (en secondes)

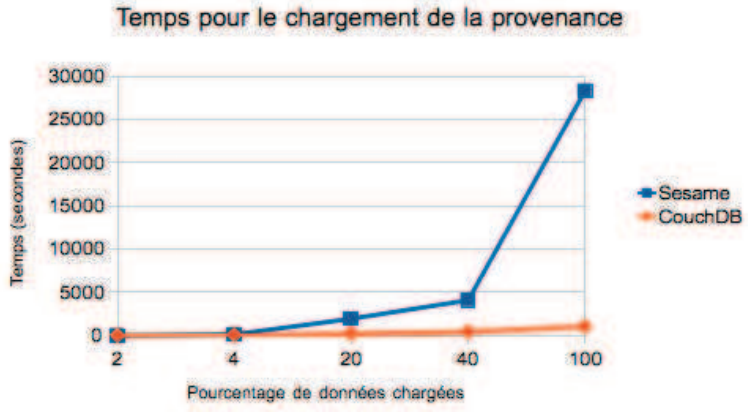


FIGURE 5.7 – Courbes de chargement de la provenance dans Sesame et CouchDB

Nous avons testé les temps de réponse pour nos six requêtes sur les deux systèmes. Les deux tables suivantes (voir Table 5.3 et Table 5.4) présentent les temps de réponse pour ces requêtes. Pour chacune d'entre elles, le résultat présenté est une moyenne du temps de réponse sur 10 exécutions consécutives.

En effectuant nos requêtes sur Sesame avec la configuration standard, nous avons rencontré des problèmes de mémoire (time-out). En augmentant la mémoire Xms Java jusqu'à 6 GO, nous avons réussi a réduire l'occurrence de ce problème, mais nous n'avons pas réussi à l'éliminer complètement : nous avons eu 4 échecs de ce type sur les 60 tests (10 exécutions pour chacune des 6 requêtes) que nous avons effectué. Les résultats de ces requêtes nous permettent de tirer les interprétations suivantes pour Sesame :

– sur les ensembles de données de petite taille (inférieure à 5 million de triplets), les temps de réponse sont satisfaisants même s'il s'agit d'une requête de chemin,
– comparées à de simples requêtes de sélection, les requêtes d'agrégation et de chemin

| | 50 K | 250 K | 1 M | 5 M | 24 M |
|---|---|---|---|---|---|
| Q1 | 26 | 36 | 49 | 63 | 439 |
| Q2 | 23 | 92 | 464 | 2463 | 13174 |
| Q3 | 51 | 52 | 55 | 61 | 173 |
| Q4 | 15 | 17 | 19 | 21 | 994 |
| Q5 | 157 | 602 | 3020 | 16014 | 85647 |
| Q6 | 155 | 593 | 2980 | 15810 | 84457 |

TABLE 5.3: Moyenne des temps d'exécution des différentes requêtes pour différent volumes de données dans Sesame (en millisecondes).

| | 50 K | 250 K | 1 M | 3 M |
|---|---|---|---|---|
| Q1 | 57 | 58 | 61 | 63 |
| Q2 | 48 | 48 | 49 | 51 |
| Q3 | 561 | 565 | 566 | 566 |
| Q4 | 55 | 55 | 57 | 58 |
| Q5 | 4106 | 4227 | 4325 | 4331 |
| Q6 | 4107 | 4231 | 4328 | 4333 |

TABLE 5.4: Moyenne des temps d'exécution des différentes requêtes pour différent volumes de données dans CouchDB (en millisecondes).

sont les plus lentes,

- toutes les requêtes dépendent du volume de données interrogées. Pour les ensembles de données de grande taille, cet impact est très remarquable surtout pour les requêtes d'agrégation et de chemin.
- le temps de réponse n'est pas linéaire et ne permettra pas de passer à l'échelle pour de gros volumes de données pour les requêtes de chemin complexes comme Q5 ou Q6.

Pour l'exécution des requêtes sur CouchDB, nous avons tiré les interprétations suivantes :

- la première requête prend 269 secondes avant de s'exécuter parce qu'elle lance le calcul de vues matérialisées sur toute la base.
- les requêtes forward et backward (Q1, Q2, Q4) sont très efficaces alors que les requêtes d'agrégation (Q3) sont plus lentes,
- les requêtes de chemin ne sont pas très efficaces mais le temps de réponse reste quand même très raisonnable,
- le temps d'exécution de toutes les requêtes est indépendant du volumes de données. Effectivement, le mécanisme de vues basé sur map/reduce est efficace sur de petits et de grands ensembles de données,
- la définition des vues nécessite une bonne maitrise du fonctionnement de map/reduce dans CouchDB. Elles sont relativement complexes pour les requêtes récursives car CouchDB ne propose pas un mécanisme intégré pour injecter des résultats intermédiaires dans une même vue.

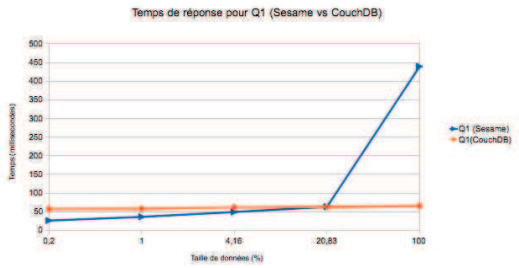


FIGURE 5.8 – Temps d'exécution de Q1 sur Sesame et CouchDB

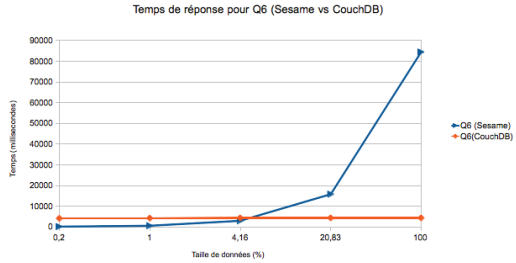


FIGURE 5.9 – Temps d'exécution de Q6 sur Sesame et CouchDB

– le temps de calcul des vues matérialisées (voir Table 5.2) est linéaire et dépend du volume de données et du type de la vue (simple ou paramétrée). L'ajout de nouvelles données à des données déjà indexées relance cette opération pour calculer de nouveau les vues matérialisées.

Ces expérimentation en chargement et en interrogation de la provenance nous permettent de tirer les conclusions suivantes :

– CouchDB assure la passage à l'échelle en stockage de données,
– pour des requêtes simples sur de faibles volumes de données ($\leq$ 2 million de triplets), Sesame est plus performant. Cependant, CouchDB assure un temps de réponse linéaire (voir constant) sur des volumes plus importants (voir Figure 5.8).
– pour les requêtes de chemin, Sesame est plus performent sur de faibles volumes de données. Cependant, CouchDB assure un temps de réponse linéaire (voir constant) sur des volumes plus importants (voir Figure 5.9).

#### 5.4.3.3 Scalabilité du module de stockage

Après avoir validé notre approche de modélisation, notre deuxième objectif est de trouver les meilleures technologies permettant la conception de systèmes de gestion de provenance scalables. Suite à nos différentes expérimentations, nous avons conclu que Sesame n'assurera pas la scalabilité souhaitée. Par contre, CouchDB semble prometteur pour atteindre ces objectifs mais possède des limites en terme d'expressivité de requêtes.

Dans cette partie, nous allons pousser encore nos tests et analyser la scalabilité de CouchDB sur des volumes de données de provenance plus importants. Pour cela, nous allons travailler sur un ensemble de données de test formé par les 2 premiers logs auxquels nous ajoutons un nouveau log du serveur Tomcat ayant pour taille 3.5 GB. Notre fonction d'import a généré environ 15 millions de documents de provenance pour ces logs que nous avons réussi à insérer sur notre serveur de test. Cependant nous avons rencontré des problèmes de mémoire pour le calcul des vues matérialisées sur cette base. Même l'augmentation de la mémoire allouée pour la machine virtuelle Erlang ne nous a pas permis de surmonter ce problème. Au lieu de configurer un seul nœud, nous proposons de

configurer un cluster CouchDB formé par plusieurs nœuds pour distribuer le stockage et les requêtes.

Par défaut, CouchDB permet de répliquer une base de données sur plusieurs nœuds mais ne propose pas nativement une configuration en mode cluster. Pour cela, nous devons passer par des solutions de clustering spécifiques comme BigCouch [18] qui est un système open source (ayant la licence Apache 2 comme CouchDB) proposé par Cloudant et permettant d'utiliser CouchDB sous forme de cluster. Il possède ainsi tous les avantages de CouchDB tout en permettant d'étendre de manière horizontale la capacité de stockage et de distribuer les documents et les bases de données. Il permet aussi une répartition du calcul des vues sur N machines. Aussi, aucun maitre n'existe dans un cluster BigCouch : chaque nœud a une connaissance des autres, ce qui permet de garantir l'accès aux données même si l'un des nœuds ne fonctionne plus. Finalement, Bigcouch se présente aux applications avec la même API que CouchDB ce qui permet de réutiliser de nombreuses librairies disponibles.

Nous avons installé un cluster de trois machines ayant une configuration identique à celle indiquée dans 5.4.1. Ensuite, nous avons créé une base distribuée et nous avons intégré respectivement 15 (ensemble D1) et 30 (ensemble D2) millions de documents. Les données ont été fragmentées horizontalement sur notre cluster (i.e on charge sur chaque nœud du cluster le même volume de données, dans notre cas on charge sur chaque nœud un tiers du volume global). A travers cette expérimentation, nous cherchons à résoudre le problème de calcul des vues matérialisées que nous rencontré sur un seul nœud. Aussi, nous voulons étudier l'impact de l'augmentation de la taille des données sur les requêtes de provenance.

Nous avons exécuté les mêmes requêtes sur notre cluster de 3 machines pour les ensembles D1 et D2. Nous présentons dans le tableau ci dessous (voir Table 5.5) le temps de réponse moyen de chaque requête pour 10 exécutions consécutives. Nous en tirons ainsi deux interprétations :

- BigCouch a permis de résoudre la problème de calcul des vues matérialisées que nous avons rencontré sur un seul nœud,
- le temps d'exécution est indépendant de la taille des données grâce à la distribution des données sur les nœuds du cluster.

Ces résultats prouvent qu'un cluster CouchDB peut assurer la scalabilité linéaire souhaitée pour le traitement des requêtes de provenance.

| Requête | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|---|
| Temps d'exécution pour D1 (15M) | 60 | 51 | 558 | 59 | 4339 | 4345 |
| Temps d'exécution pour D2 (30M) | 63 | 53 | 560 | 60 | 4340 | 4347 |

TABLE 5.5: Temps moyen d'exécution des requêtes sur BigCouch pour les ensembles D1 et D2 (en millisecondes).

La dernière expérimentation est dédiée à la scalabilité des requêtes de chemin. Ce

18. http ://bigcouch.Cloudant.com

type de requête calcule de manière récursive un ensemble de résultats intermédiaires pour identifier un chemin. Pour tous les autres types de requêtes, le temps de réponse est resté indépendant de l'augmentation du volume de données à interroger. Nous souhaitons évaluer l'impact de la taille des résultats intermédiaires sur le temps d'exécution des requêtes de chemin dans CouchDB et vérifier si le volume de données n'a pas une influence sur leur temps d'exécution. Nous effectuons ce test sur la requête Q6 et sur trois ensembles de données : D'1 (3 millions de documents de provenance), D'2 (15 millions de documents de provenance) et D'3 (30 millions de documents de provenance). Pour chacun de ces ensembles, nous effectuons des requêtes faisant intervenir 6, 15 et 20 résultats intermédiaires (en modifiant l'identifiant du document interrogé).

Les résultats des tests effectués sur D'1, D'2 et D'3 pour Q6 (temps d'exécution moyen sur 10 exécutions) sont présentés ci-dessous (Table 5.6). Ils prouvent bien que le nombre de résultats intermédiaires a un impact important sur le temps de réponse : une augmentation de 333% du nombre de résultats intermédiaires implique une augmentation de 215 % environ du temps d'exécution moyen (voir Figure 5.10 ).

| Nombre de résultats intermédiaires | 6 | 15 | 20 |
|---|---|---|---|
| Temps d'exécution pour D'1 | 4285 | 6829 | 9374 |
| Temps d'exécution pour D'2 | 4343 | 6835 | 9387 |
| Temps d'exécution pour D'3 | 4359 | 6829 | 9397 |

TABLE 5.6: Impact de la taille des résultats intermédiaires sur le temps d'exécution de Q6 (en millisecondes).

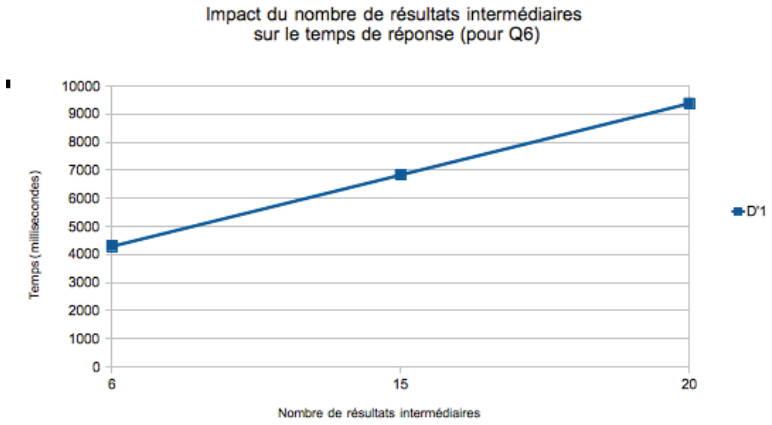


FIGURE 5.10 – Impact du nombre de résultats intermédiaires sur les requêtes de chemin (pour l'ensemble D'1)

### 5.4.4 Synthèse

Dans cette section, nous avons présenté une validation non fonctionnelle de notre travail en proposant deux prototypes de PMS basés sur deux technologies de stockage différentes. Le premier prototype, basé sur le RDF store Sesame possède l'avantage de supporter le langage de requête SPARQL conçu pour le Web sémantique et les données liées. Le prototype de PMS que nous avons implémenté sur Sesame a montré des limites très contraignantes pour le chargement et l'interrogation de gros volumes de données de provenance.

Le deuxième prototype basé sur CouchDB, un système NoSQL orienté document nous a permis de charger des volumes de données de provenance très importants et de les interroger avec un temps linéaire grâce à son mécanisme de requêtes basé sur map/reduce. Nous avons ensuite fait évoluer ce deuxième prototype vers un système plus scalable en configurant un cluster basé sur BigCouch. Ce cluster assure d'étendre horizontalement la capacité de stockage et de distribuer les requêtes sur les différents nœuds. Nos expérimentations montrent que ce prototype assure un temps de réponse acceptable et linéaire sur tous les types de requêtes que nous avons présenté.

## 5.5 Conclusion

Ce chapitre a été consacré à la validation de notre approche de modélisation d'un point de vue fonctionnel, à la présentation de services de tracabilité développés pour Novapost et à la validation non-fonctionnelle de la scalabilité de notre architecture de PMS.

Notre approche de modélisation de provenance basée sur le framework sémantique que nous avons présenté dans le chapitre 3 a été utilisée pour proposer un modèle de provenance pour la facturation dans un environnement de Cloud computing. Cette modélisation, basée sur la spécialisation du $MDM$, montre que notre approche de modélisation est générique est peut satisfaire des besoins issus d'un domaine différent de celui de la gestion documentaire et de l'archivage légal. Des requêtes qui portent sur la provenance d'applications/services dans le Cloud et sur les détails de leurs coûts peuvent être formalisées pour répondre aux besoins de facturation des administrateurs d'un Cloud broker.

Ensuite, nous avons présenté dans la partie 5.3 notre travail de développement qui porte sur l'implémentation de services REST pour la génération, l'enrichissement et la consultation de données de provenance relatives aux processus de gestion documentaires chez Novapost (processus de traitement, d'archivage et de distribution de documents). Comme le développement de ces services a précédé la mise en place complète de notre framework, nous avons montré comment interfacer les développements effectués dans notre architecture de PMS.

Enfin, pour valider la faisabilité de notre approche pour la conception de PMS, nous avons mis en place deux prototypes de systèmes de gestion de provenance basés sur deux technologies de stockage différentes. Le premier prototype utilise un RDF store (Sesame)

et permet d'utiliser un langage de requête sémantique, déclaratif et expressif (SPARQL). Cependant, ce type de stockage a montré des limites qui empêchent la sclabalité de l'architecture sur de grands volumes de données de provenance. Bien que l'utilisation d'une telle technologie (RDF store) s'inscrit dans notre logique d'utilisation des technologie du Web sémantique pour la modélisation et la conception de PMSs, nos expérimentation prouvent que ces systèmes sont aujourd'hui incapables d'assurer la scalabilité que nous souhaitons. Pour pallier à cette limite, nous avons présenté les caractéristiques des SGBDs NoSQL que nous jugeons adaptées à la structure de la provenance et aux exigences de passage à l'échelle de PMSs pour le stockage et l'interrogation de gros volumes de données de provenance. Aver sa structure orientée document extensible, CouchDB nous a permis d'utiliser notre approche de modélisation et d'assurer une scalabilité et un temps de réponse linéaire pour tous les types de requêtes de provenance, ce qui n'est pas le cas pour Sesame. Des tests de passage à l'échelle sur des volumes plus importants ont été également effectués sur un cluster BigCouch de trois machines. Sur ce cluster, les temps de réponse aux requêtes étaient en cohérence avec nos premières interprétations pour le passage à l'échelle sur un seul nœud CouchDB. Cette linéarité du temps de réponse est expliquée par l'utilisation de map/reduce qui assure la distribution des requêtes sur tous les nœuds du cluster. Ce travail d'expérimentation et de prototypage a été validé par une publication dans une conférence internationale [Sakka et Defude, 2012b].

*Chapitre 6*

# Conclusions et perspectives

*"C'est là en effet un des grands et merveilleux caractères des beaux livres que pour l'auteur ils pourraient s'appeler Conclusions et pour le lecteur Incitations."*

Marcel Proust, "Sur la lecture"

## 6.1 Conclusions

L'objectif principal de ce travail de thèse était de proposer une approche pour la modélisation de la provenance et une architecture de PMS mettant en œuvre cette approche de modélisation et qui soit capable de gérer l'intégralité du cycle de vie de la provenance dans un contexte multi-sources distribué. Pour atteindre cet objectif, nous avons proposé une approche pour la modélisation et la gestion de la provenance basée sur les technologies du Web sémantique, une architecture logique pour un système de gestion de provenance mettant en œuvre cette approche de modélisation et une architecture distribuée d'un PMS à base de médiateur préservant l'autonomie de sources de provenance distribuées.

L'approche de modélisation que nous proposons se distingue des approches existantes par le fait qu'elle propose une démarche globale et cohérente pour la modélisation et la gestion de provenance. Elle couvre les différents aspects de collecte, gestion, stockage, corrélation et interrogation de la provenance.

Notre approche se base sur un modèle générique, extensible, assurant l'interopérabilité syntaxique ainsi que interopérabilité sémantique entre des modèles de provenance hétérogènes. En effet, notre approche propose un modèle de domaine minimal (MDM) basé sur le modèle OPM qui est un modèle de référence pour la modélisation de provenance. Ce modèle minimal garantit la généricité, l'extensibilité et l'interopérabilité syntaxique entre les modèles de provenance. Ensuite, notre approche propose de capitaliser les concepts et les propriétés d'un domaine métier dans des ontologies que nous appelons modèles de domaine (MD). Ces modèles de domaine sont définis par spécialisation du modèle MDM ce qui garantit l'interopérabilité sémantique souhaitée.

Pour l'exploitation de la provenance selon plusieurs contraintes et visions métier, nous avons proposé la création de vues métier. Ces vues métier peuvent être générées automatiquement via un processus de généralisation. Ce processus consiste à créer, à partir d'un modèle de domaine instancié (correspondant à un modèle d'import) un treillis de modèles de domaines instanciées correspondant à des modèles de domaine plus génériques que le modèle d'import. Nous obtenons ainsi une treillis de modèles de domaine instanciés correspondant à des vues métier dont la borne inférieure est le modèle d'import instancié et la borne supérieure est le modèle de domaine minimal.

Ensuite, nous nous sommes intéressés à la conception de systèmes de gestion de provenance. Pour concevoir des PMSs couvrant la collecte, la gestion, la corrélation et l'interrogation de la provenance, nous avons proposé de mettre en oœuvre notre approche de modélisation dans une architecture logique. Cette architecture est indépendante des choix technologiques d'implémentation et de déploiement. Elle se compose d'un module de collecte, d'un module pour la gestion de modèles de domaines, d'un module pour la gestion des modèles de domaine instanciés, d'un module pour la gestion de fusion, d'un module de stockage ainsi qu'un module pour le traitement de requêtes. Elle sert comme une architecture de référence pour la conception de PMSs exploitant des données de provenance existantes (comme les logs applicatifs) et ayant des fonctionnalités permettant de pallier à l'hétérogénéité syntaxique et sémantique entre les modèles.

Afin de préserver l'autonomie inter-sources de provenance, nous avons proposé une architecture distribuée de PMS à base de médiateur. Ce médiateur possède une vision globale sur l'ensemble des sources de provenance ainsi que des capacités de traitement et de distribution de requêtes. Il effectue un ensemble d'optimisations assurant que la requête sera traitée par un ensemble minimal et complet de sources. L'autre avantage de cette architecture est que le médiateur possède une capacité de corrélation entre les identifiants d'artifacts distribués ce qui permet d'inférer de nouvelles connaissances et d'obtenir des résultats pertinents.

Nous avons validé fonctionnellement notre approche de modélisation sur des données de provenance de Novapost avec des modèles métier d'archivage légal et de gestion électronique de documents. La validation fonctionnelle de la généricité de notre approche de modélisation a été effectuée en modélisant les besoins de provenance dans un contexte de Cloud computing basé sur un Cloud broker. Ensuite, nous avons effectué la validation non-fonctionnelle de notre architecture sur les aspects de passage à l'échelle. Cette validation repose sur l'implémentation de deux prototypes de PMS sur deux technologies de stockage différentes : un système RDF (Sesame) et un SGBD NoSQL (CouchDB). Les tests de montée en charge effectués sur les données de provenance Novapost ont montré les limites de Sesame tant pour le stockage que pour l'interrogation en utilisant le langage de requêtes SPARQL, alors que la version CouchDB associée à un langage de requêtes basé sur map/reduce a démontré sa capacité à suivre la charge de manière linéaire en augmentant le nombre de serveurs. Cette validation nous a permis de confirmer qu'il est possible de garder l'expressivité

sémantique de notre modèle en utilisant un SGBD NoSQL. Elle nous a confirmé aussi qu'il est possible de formaliser des requêtes de provenance complexes mettant en œuvre des propriétés métier sur ce type de SGBDs. Cependant, la formalisation de ces requêtes est plus complexe et moins déclarative que SPARQL.

A travers les différentes contributions proposées, nous avons résolu la problématique de recherche exprimée au début de notre thèse. En effet, nous avons proposé une approche de modélisation de la provenance basée sur les technologies du Web sémantique permettant de pallier à l'hétérogénéité syntaxique et sémantique entre les modèles de provenance.

Dans notre approche, nous avons traité la gestion de l'aspect multi-sources, la fusion de modèles de domaine hétérogènes ainsi que la corrélation d'identifiants distribués. Ceci permet d'exploiter des logs hétérogènes et d'effectuer des inférences permettant d'identifier les cycles de vie de documents à travers les applications. Ces aspects n'étaient pas développés dans aucun travail de l'état de l'art. Leur traitement constitue l'un des points forts de notre approche de modélisation.

En se basant sur cette approche, nous avons proposé une architecture logique de PMS et une architecture de PMS distribué à base de médiateur garantissant l'autonomie inter-PMSs. L'avantage de l'architecture distribuée du PMS proposé est qu'elle tire profit de l'interopérabilité de notre modèle. Elle se base sur la duplication des instances du MDM et permet de garder les applications sources indépendantes les unes des autres. Elle est très proche des systèmes LAV, cependant, il ne s'agit pas d'une médiation pure mais plutôt d'une médiation basée sur les schémas au niveau des sources ainsi que les instances dupliquées sur le médiateur ($MDM_{instancie}$).

L'implémentation de deux PMS dont nous avons testé le passage à l'échelle en stockage et en interrogation valident l'intérêt et la faisabilité de notre approche et se différencient de la majorité des PMSs de l'état de l'art qui ne traitent pas cet aspect. En plus, l'utilisation d'un SGBD NoSQL pour le stockage de la provenance est une proposition intéressante et très bien adaptée à la nature et aux caractéristiques de la provenance.

## 6.2 Perspectives

Dans cette section, nous présentons quelques directions de nos futurs travaux. Il s'agit des perspectives liées à nos principales contributions le long de cette thèse : la modélisation de la provenance et la conception d'architectures de PMSs scalables.

### 6.2.1 Perspectives liées à la modélisation de la provenance

Nous avons proposé dans notre approche la gestion de sources de provenance multiples via une approche de fusion. Cette fusion vérifie la compatibilité sémantique entre les modèles à fusionner ainsi que la cohérence entre les données. Ce processus vise à corréler des artifacts distribués et de générer de nouvelles connaissances qui étaient non explicites si les sources de provenance étaient gérées séparément. Nous devons aller plus loin dans

la réalisation de ce processus qui se base actuellement sur les identifiants d'artifacts et identifier de nouveaux types d'inférences qui pourront assurer la pertinence et la complétude de la chaine de provenance.

Nous envisageons aussi de valider la généricité de notre approche sur d'autres types d'applications et contextes d'utilisation.

Également, nous envisageons d'approfondir notre travail sur la partie fusion des $MDIs$. L'idée est de proposer un outil d'assistance à l'administrateur lui permettant de choisir la meilleure transformation lorsque les $MDs$ ne sont pas compatibles. En effet, dans ce cas plusieurs transformations sont possibles. Parmi les critères de choix intéressants, on peut chercher à minimiser le nombre de concepts, propriétés et associations perdues tant au niveau des $MDs$ que des $MDIs$.

### 6.2.2 Perspectives liées à la conception de PMSs

Dans le chapitre 4, nous avons proposé une algorithme de distribution implémenté au niveau du médiateur. Les tests de passage à l'échelle sur notre PMS basé sur CouchDB et map/reduce nous ont permis de vérifier que le système suit la charge de manière linéaire en augmentant le nombre de serveurs. Il reste à le vérifier sur un nombre beaucoup plus important de nœuds (quelques dizaines). Par ailleurs, le rendement d'un nœud est assez faible et il serait intéressant de l'améliorer.

Un autre perspective consiste à l'implémentation de notre algorithme de distribution et son déploiement dans un environnement industriel. Nous voulons valider le fait que la mise en place d'une telle architecture ne demande pas un grand travail d'intégration du coté des systèmes sources. Ensuite, nous voulons tester notre approche de distribution et comparer les temps de réponse aux requêtes pour ce système (la version distribuée) avec les temps de réponses obtenus dans le chapitre 5 sur la version de PMS centralisé.

Nous visons aussi d'approfondir encore notre algorithme d'évaluation distribuée de requêtes. Il s'agit de vérifier qu'il est capable de traiter toutes les catégories de requêtes de provenance et d'étudier les différent types de réécriture/optimisations possibles. Dans ce même contexte d'interrogation, nous jugeons utile d'approfondir encore la technique de corrélation que nous avons proposé et d'identifier les inférences possibles et s'il est possible de les améliorer en modifiant le modèle de données.

Nous envisageons aussi de tester notre approche de distribution dans un contexte hétérogène où les PMSs sources n'implémentent pas notre framework et utilisent des modèles différents qui ne sont pas issus d'une spécialisation du MDM. Dans ce cas, nous devons mettre en place des wrappers pour convertir une partie des données vers le modèle MDM afin de les stoker sur le médiateur ainsi que des techniques de réécriture de requêtes permettant de réécrire les requêtes issues du médiateur dans des langages spécifiques aux PMSs correspondants.

Sur le long terme, nous envisageons d'intégrer des techniques de sécurisation de la provenance dans notre approche de modélisation. Le but est de mettre en place des techniques

permettent de générer des données de provenance sûres et de garantir leur intégrité et leur confidentialité. Par exemple, le travail proposé par Hasan et al. [Hasan *et al.*, 2009] propose les fondations nécessaires pour la sécurisation de la provenance dans des systèmes monolithiques (comme le noyau d'un système d'exploitation par exemple). Nous envisageons la mise en place et le développement des schémas de sécurisation proposés par Hasan et al. sur des systèmes ouverts et distribués pour chainer les informations de provenance et leur attacher des méta-données de sécurisation dès leur création.

# Publications de l'auteur

## Conférences internationales avec comité de lecture et avec actes

1. Mohamed Amin Sakka and Bruno Defude. "Scalability Issues in Designing and Implementing Semantic Provenance Management Systems". In the proceedings of the 5th International Conference on Data Management in Cloud, Grid and P2P Systems (Globe 2012), Vienna, Austria, September 2012 (To appear).
2. Mohamed Amin Sakka and Bruno Defude. "A Mediator-based System for Distributed Semantic Provenance Management Systems". In the proceedings of the 16th International Database Engineering & Applications Symposium, Prague, Czech Republic, Août 2012 (To appear).
3. Mohamed Amin Sakka, Bruno Defude and Jorge Tellez. "A Semantic Framework for the Management of Enriched Provenance Logs". In the proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA 2012), Fukuoka, Japon, Mars 2012.
4. Mohamed Amin Sakka, Bruno Defude and Jorge Tellez. "Document provenance in the cloud : constraints and challenges". In the proceedings of the 16th EUNICE/IFIP WG 6.6 conference on Networked services and applications (EUNICE 2010), Trondheim, Norway, June 2010.

# Bibliographie


[Abadi *et al.*, 2007] Abadi, D. J., Marcus, A., Madden, S. R. et Hollenbach, K. (2007). Scalable semantic web data management using vertical partitioning. *In Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 411–422. VLDB Endowment. [cité p. 30]

[AFNOR, 2009] AFNOR (2009). Archivage électronique - Spécifications relatives à la conception et à l'exploitation de systèmes informatiques en vue d'assurer la conservation et l'intégrité des documents stockés dans ces systèmes (NF Z42-013). [cité p. 41, 48]

[Bailey *et al.*, 2005] Bailey, J., Bry, F., Furche, T. et Schaffert, S. (2005). Web and Semantic Web Query Languages : A Survey. *Reasoning Web*, pages 35–133. [cité p. 117]

[Bechhofer *et al.*, 2004] Bechhofer, S., Harmelen, F. V., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. et Andrea, S. L. (2004). OWL Web Ontology Language Reference. [cité p. 36]

[Berners-Lee, 1998] Berners-Lee, T. (1998). Semantic Web Road map. Octobre 1998 http://www.w3.org/DesignIssues/Semantic.html. [cité p. 30]

[Berners-Lee *et al.*, 2001] Berners-Lee, T., Hendler, J. et Lassila, O. (2001). The Semantic Web : Scientific American. *Scientific American*. [cité p. 30, 31]

[Buneman *et al.*, 2000] Buneman, P., Khanna, S. et chiew Tan, W. (2000). Data provenance : some basic issues. *In In Foundations of Software Technology and Theoretical Computer Science*, pages 87–93. Springer. [cité p. 20, 48]

[Buneman *et al.*, 2001] Buneman, P., Khanna, S. et Tan, W.-C. (2001). Why and Where : A Characterization of Data Provenance. *In In ICDT*, pages 316–330. [cité p. 20]

[Cao *et al.*, 2009] Cao, B., Plale, B., Subramanian, G., Robertson, E. et Simmhan, Y. (2009). Provenance Information Model of Karma Version 3. *In Proceedings of the 2009 Congress on Services - I*, pages 348–351, Washington, DC, USA. IEEE Computer Society. [cité p. 22]

[Cattell, 2011] Cattell, R. (2011). Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39(4):12–27. [cité p. 121]

[Chang *et al.*, 2006] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A. et Gruber, R. E. (2006). Bigtable : A distributed storage system for structured data. *In IN PROCEEDINGS OF THE 7TH CONFERENCE ON*

*USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION - VOLUME 7*, pages 205–218. [cité p. 121]

[Chebotko *et al.*, 2010] Chebotko, A., Lu, S., Fei, X. et Fotouhi, F. (2010). RDFProv : A relational RDF store for querying and managing scientific workflow provenance. *Data and Knowledge Engineering*, pages 836–865. [cité p. 28, 110]

[Dean et Ghemawat, 2010] Dean, J. et Ghemawat, S. (2010). MapReduce : a flexible data processing tool. *Commun. ACM*, 53(1):72–77. [cité p. 121]

[Euzenat et Shvaiko, 2007] Euzenat, J. et Shvaiko, P. (2007). *Ontology matching*. Springer-Verlag, Heidelberg (DE). [cité p. 68]

[Fensel *et al.*, 2000] Fensel, D., Horrocks, I., Harmelen, F. V., Decker, S., Erdmann, M. et Klein, M. (2000). OIL in a nutshell. pages 1–16. [cité p. 34]

[Fielding, 2000] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. Thèse de doctorat, University of California, Irvine. [cité p. 123]

[Freitas *et al.*, 2012] Freitas, A., Curry, E., Oliveira, J. G. et O'Riain, S. (2012). Querying Heterogeneous Datasets on the Linked Data Web : Challenges, Approaches, and Trends. *IEEE Internet Computing*, 16:24–33. [cité p. 86]

[Freitas *et al.*, 2010] Freitas, A., Legendre, A., O'Riain, S. et Curry, E. (2010). Prov4J : A Semantic Web Framework for Generic Provenance Management. *In The Second International Workshop on Role of Semantic Web in Provenance Management,SWPM 2010*. [cité p. 30, 38, 39, 45]

[Gorlitz et Staab, 2011] Gorlitz, O. et Staab, S. (2011). Federated Data Management and Query Optimization for Linked Open Data. *Data Management*, pages 109–137. [cité p. 86]

[Groth, 2010] Groth, P. (2010). ProvenanceJS : Revealing the Provenance of Web Pages. *In Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers*, volume 6378 de *Lecture Notes in Computer Science*, pages 283–285. Springer. [cité p. 22]

[Groth *et al.*, 2006] Groth, P., Jiang, S., Miles, S., Munroe, S., Tan, V., Tsasakou, S. et Moreau, L. (2006). An Architecture for Provenance Systems. Rapport technique. http://eprints.ecs.soton.ac.uk/13196. [cité p. 27, 29, 40, 45, 47]

[Groth *et al.*, 2004] Groth, P., Luck, M. et Moreau, L. (2004). A protocol for recording provenance in service-oriented grids. *In In Proceedings of the 8th International Conference on Principles of Distributed Systems(OPODIS 2004*, pages 124–139. [cité p. 27, 45]

[Groth *et al.*, 2009] Groth, P., Miles, S. et Moreau, L. (2009). A model of process documentation to determine provenance in mash-ups. *ACM Trans. Internet Technol.*, 9:3 :1–3 :31. [cité p. 83]

[Groth et Moreau, 2011] Groth, P. et Moreau, L. (2011). Representing distributed systems using the Open Provenance Model. *Future Gener. Comput. Syst.*, 27(6):757–765. [cité p. 110]

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220. [cité p. 34]

[Hartig, 2009] Hartig, O. (2009). Provenance Information in the Web of Data. *In Second Workshop on Linked Data on the Web, LDOW*. [cité p. 30, 40, 45]

[Hartig et Zhao, 2010] Hartig, O. et Zhao, J. (2010). Publishing and Consuming Provenance Metadata on the Web of Linked Data. *In IPAW*, pages 78–90. [cité p. 40, 41, 47]

[Hasan *et al.*, 2009] Hasan, R., Sion, R. et Winslett, M. (2009). Preventing history forgery with secure provenance. *ACM Transactions on Storage*, 5:12 :1–12 :43. [cité p. 43, 44, 45, 47, 139]

[Hiep, 2007] Hiep, L. P. (2007). *Gestion de l'évolution d'un Web sémantique d'entreprise*. Thése de doctorat, École Nationale Supérieure des Mines de Paris. [cité p. 35]

[ISO15489-1, 2001] ISO15489-1 (2001). Norme ISO 15489-1 : Information et documentation Records management Partie 1 : Principes directeurs. Diponible sur : www.iso.org/iso/fr/catalogue_detail?csnumber=31908. [cité p. 16]

[Laublet *et al.*, 2002] Laublet, P., Reynaud, C. et Charlet, J. (2002). Sur quelques aspects du web sémantique. *In Actes des deuxiémes assises du GdR 13, pp 59-78, Cépaduès Editions*. [cité p. 34]

[Li et Boucelma, 2011] Li, Y. et Boucelma, O. (2011). A CPN Provenance Model of Workflow : Towards Diagnosis in the Cloud. *In Proceedings of the 15th East-European Conference on Advances in Databases and Information Systems (ADBIS 2011)*, pages 55–64. [cité p. 110]

[Moreau, 2006] Moreau, L. (2006). Usage of provenance : A Tower of Babel Towards a concept map. Position paper for the Microsoft life cycle seminar. Rapport technique, University of Southampton. [cité p. 17]

[Moreau, 2010] Moreau, L. (2010). The Foundations for Provenance on the Web. *Foundations and Trends in Web Science*, 2:99–241. [cité p. 56]

[Moreau *et al.*, 2010a] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y. L., Stephan, E. et Bussche, J. V. (2010a). The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*. [cité p. 22, 23]

[Moreau *et al.*, 2008] Moreau, L., Freire, J., Futrelle, J., Mcgrath, R., Myers, J. et Paulson, P. (2008). The Open Provenance Model : An Overview. pages 323–326. [cité p. 55]

[Moreau *et al.*, 2010b] Moreau, L., Groth, P. et Zhao, J. (2010b). Open Provenance Model Tutorial Session : OPM Emerging Profiles. *In Future Internet Symposium*. [cité p. 110]

[MoReq2, 2008] MoReq2, E. C. (2008). Model Requirements for the Management of Electronic Records (MoReq2) Specification. Accessible sur wwww.moreq.eu. [cité p. 41, 48]

[Muniswamy-Reddy et Seltzer, 2010] Muniswamy-Reddy, K. K. et Seltzer, M. (2010). Provenance as first class cloud data. *SIGOPS Oper. Syst. Rev.*, 43:11–16. [cité p. 19]

[Ozsu et Valduriez, 2011] Ozsu, M. T. et Valduriez, P. (2011). *Principles of Distributed Database Systems*. Springer-Verlag, Berlin, Heidelberg, 3Őme édition. [cité p. 44, 85]

[Pavlo *et al.*, 2009] Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D., Madden, S. et Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. *In Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 165–178, New York, NY, USA. ACM. [cité p. 122]

[Prud'hommeaux et Seaborne, 2008] Prud'hommeaux, E. et Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation. Accessible sur http://www.w3.org/TR/rdf-sparql-query. [cité p. 36, 90]

[Quilitz et Leser, 2008] Quilitz, B. et Leser, U. (2008). Querying Distributed RDF Data Sources with SPARQL. *In* Bechhofer, S., Hauswirth, M., Hoffmann, J. et Koubarakis, M., éditeurs : *The Semantic Web : Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 de *Lecture Notes in Computer Science*, pages 524–538. Springer. [cité p. 86]

[Ram et Liu, 2009] Ram, S. et Liu, J. (2009). A New Perspective on Semantics of Data Provenance. *In The First International Workshop on Role of Semantic Web in Provenance Management, SWPM 2009*. [cité p. 20, 23]

[Sahoo *et al.*, 2009] Sahoo, S. S., Barga, R., Sheth, A., Thirunarayan, K. et Hitzler, P. (2009). PrOM : A Semantic Web Framework for Provenance Management in Science. Rapport technique KNOESIS-TR-2009, Kno.e.sis Center. [cité p. 30, 37, 38, 45, 78]

[Sahoo *et al.*, 2008] Sahoo, S. S., ILMand, Sheth, A. et Henson, C. (2008). Semantic Provenance for eScience : Managing the Deluge of Scientific Data. *IEEE Internet Computing*, 12:46–54. [cité p. 37]

[Sakka et Defude, 2012a] Sakka, M. A. et Defude, B. (2012a). A mediator-based system for distributed semantic provenance management systems. *In Proc. of the 16th International Database Engineering and Applications Symposium*, Prague, Czech Republic. [cité p. 105]

[Sakka et Defude, 2012b] Sakka, M. A. et Defude, B. (2012b). Scalability Issues in Designing and Implementing Semantic Provenance Management Systems. *In Proc. of the 5th International Conference on Data Management in Cloud, Grid and P2P Systems (Globe 2012)*, Vienna, Austria. [cité p. 134]

[Sakka *et al.*, 2012] Sakka, M. A., Defude, B. et Tellez, J. (2012). A Semantic Framework for the Management of Enriched Provenance Logs. *In Proc. of the 26th IEEE International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan. IEEE Computer Society. [cité p. 75]

[Sidirourgos *et al.*, 2008] Sidirourgos, L., Goncalves, R., Kersten, M., Nes, N. et Manegold, S. (2008). Column-store support for RDF data management : not all swans are white. *Proc. VLDB Endow.*, 1(2):1553–1563. [cité p. 30]

[Silva *et al.*, 2006] Silva, P. P. D., L.McGuinness, D. et Fikes, R. (2006). A proof markup language for Semantic Web services. *Information. Systems*, 31:381–395. [cité p. 40]

[Simmhan, 2009] Simmhan, Y. L. (2009). Feedback On OPM. Décembre 2011 : http://twiki.ipaw.info/pub/Challenge/OpenProvenanceModelWorkshop/FeedbackonOPM.pptx. [cité p. 23, 59]

[Sivasubramanian, 2012] Sivasubramanian, S. (2012). Amazon dynamoDB : a seamlessly scalable non-relational database service. *In Proceedings of the 2012 international conference on Management of Data*, SIGMOD '12, pages 729–730, New York, NY, USA. ACM. [cité p. 122]

[SNIA, 2004] SNIA (2004). Information Lifecycle Management : Vision for ILM. Disponible sur : www.snia.org/forums/dmf/programs/ilmi/ilm_docs/DMF_slides_for_SNIA_KIOSK_at_SNW_April_2004.pdf. [cité p. 20]

[Sowa, 1984] Sowa, J. (1984). *Conceptual Structures : Information Processing in Mind and Machine*. The Systems Programming Series. Addison-Wesley. [cité p. 21]

[Stevens *et al.*, 2003] Stevens, R. D., Robinson, A. J. et Goble, C. (2003). myGrid : personalised bioinformatics on the information grid. *Bioinformatics*, 19 Suppl 1. [cité p. 26, 45]

[Stonebraker *et al.*, 2010] Stonebraker, M., Abadi, D. J., DeWitt, D., Madden, S., Paulson, E., Pavlo, A. et Rasin, A. (2010). MapReduce and parallel DBMSs : friends or foes ? *Commun. ACM*, 53(1):64–71. [cité p. 122]

[Strauch, 2010] Strauch, C. (2010). NoSQL Databases. *Lecture Notes Stuttgart Media*. [cité p. 121, 122]

[UKCRC, 2008] UKCRC (2008). Grand Challenges in Computing Research Conference. Disponible sur : www.ukcrc.org.uk/press/news/challenge08/gccr08final.cfm. [cité p. 19]

[Widom, 2005] Widom, J. (2005). Trio : A System for Integrated Management of Data, Accuracy, and Lineage. *In Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*. [cité p. 25, 26, 45]

[Zhao, 2007] Zhao, J. (2007). *A conceptual model for e-science provenance*. Thèse de doctorat, University of Manchester. [cité p. 40]

[Zhao *et al.*, 2004] Zhao, J., Goble, C., Stevens, R. et Bechhofer, S. (2004). Semantically Linking and Browsing Provenance Logs for E-science. *In Proc. of the 1st International IFIP Conference on Semantics of a Networked World (ICSNW)*, pages 158–176. [cité p. 26]

[Zhao *et al.*, 2011a] Zhao, J., Sahoo, S. S., Missier, P., Sheth, A. et Goble, C. (2011a). Extending Semantic Provenance into the Web of Data. *IEEE Internet Computing*, 15:40–48. [cité p. 78]

[Zhao *et al.*, 2011b] Zhao, J., Simmhan, Y., Gomadam, K. et Prasanna, V. K. (2011b). Querying Provenance Information in Distributed Environments. *International Journal of Computers and Their Applications (IJCA)*, 18(3):196–215. [cité p. 44, 45, 47, 78]